Πληροφορική & Τηλεπικοινωνίες

Κ18 - Υλοποίηση Συστημάτων Βάσεων Δεδομένων
Χειμερινό Εξάμηνο 2015 – 2016

Ι. Ιωαννίδης

Άσκηση 3: Αποθήκευση κατά στήλες
Προθεσμία: 25 Ιανουαρίου 2016, 11:59μμ

Ο σκοπός της εργασίας αυτής είναι η κατανόηση των Συστημάτων Διαχείρισης Βάσεων Δεδομένων που βασίζονται στην τεχνική αποθήκευση κατά στήλες (column store) για την διαχείριση των δεδομένων. Για πληροφορίες σχετικά με column stores, σας προτείνουμε να δείτε και τα παρακάτω:

- https://en.wikipedia.org/wiki/Column-oriented_DBMS
- http://nms.csail.mit.edu/~stavros/pubs/tutorial2009-column_stores.pdf

Η διαχείριση των αρχείων γίνεται μέσω των συναρτήσεων με το πρόθεμα CS_ (column store). Τις εγγραφές τις διαχειρίζεστε μέσω των κατάλληλων συναρτήσεων. Τα πρωτότυπα (definitions) των συναρτήσεων που καλείστε να υλοποιήσετε δίνονται στη συνέχεια μαζί με επεξήγηση για τη λειτουργικότητα της κάθε μίας. Όπως και στην προηγούμενη άσκηση, οι συναρτήσεις της βιβλιοθήκης επιπέδου μπλοκ θα σας δίνονται έτοιμες.

```
typedef struct {
        int id,
        char name[15],
        char surname[20],
        char status[1],
        char dateOfBirth[11],
        int salary;
} Record;
```

Μία από τις βασικές διαφορές των column stores είναι ο τρόπος αποθήκευσης των εγγραφών. Στις κλασικές βάσεις δεδομένων, όλα τα πεδία μιας εγγραφής αποθηκεύονται στο ίδιο αρχείο. Στα column stores, το κάθε πεδίο μιας εγγραφής αποθηκεύετε σε διαφορετικό αρχείο. Στα πλαίσια της άσκησης, το column-store για τα 6 πεδία της εγγραφής Record, θα δημιουργεί 6 αρχεία, στα οποία θα αποθηκεύονται οι εγγραφές: id, name, surname, status, dateOfBirth, salary αντίστοιχα σε κάθε αρχείο.

Το πρώτο μπλοκ (block) κάθε αρχείου, περιλαμβάνει “ειδική” πληροφορία σχετικά με το ίδιο το αρχείο. Η πληροφορία αυτή χρησιμεύει στο να αναγνωρίσει κανείς το είδος του αρχείου, σε πληροφορία που αφορά το πεδίο κ.λπ.

Στη συνέχεια δίνεται ένα παράδειγμα των εγγραφών που θα προστίθενται στα αρχεία που θα δημιουργείτε με την υλοποίησή σας.

{22, “Vaggelis”, “Nikolopoulos”, “S”, “02-02-1981”, 2000}
{7, “Christoforos”, “Svingos”, “M”, “28-01-1983”, 2100}
{86, “Yannis”, “Chronis”, “S”, “06-04-1965”, 3500}

Συναρτήσεις BF (Block File)

Στη συνέχεια, περιγράφονται οι συναρτήσεις που αφορούν το επίπεδο από block, πάνω στο οποίο θα βασιστείτε για την υλοποίηση των συναρτήσεων που ζητούνται όπως και στις προηγούμενες ασκήσεις. Η υλοποίηση των συναρτήσεων αυτών θα δοθεί έτοιμη με τη μορφή βιβλιοθήκης. Στο αρχείο κεφαλίδας BF.h που θα σας δοθεί υπάρχουν τα πρωτότυπα των συναρτήσεων μαζί με σχόλια για το πώς δουλεύουν και το μέγεθος ενός μπλοκ που είναι BF_BLOCK_SIZE, ίσο με 1024 bytes.

void BF_Init()

Με τη συνάρτηση BF_Init πραγματοποιείται η αρχικοποίηση του επιπέδου BF.

int BF_CreateFile(char* filename /* όνομα αρχείου */)

Η συνάρτηση BF_CreateFile δημιουργεί ένα αρχείο με όνομα filename το οποίο αποτελείται από block. Αν το αρχείο υπάρχει ήδη το παλιό αρχείο διαγράφεται. Σε περίπτωση επιτυχούς εκτέλεσης της συνάρτησης επιστρέφεται 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση BF_PrintError.

int BF_OpenFile(char* filename /* όνομα αρχείου */)

Η συνάρτηση BF_OpenFile ανοίγει ένα υπάρχον αρχείο από block με όνομα filename. Σε περίπτωση επιτυχίας, επιστρέφεται το αναγνωριστικό του αρχείου, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση BF_PrintError.

int BF_CloseFile(
int fileDesc /* αναγνωριστικό αρχείου block */)

Η συνάρτηση BF_CloseFile κλείνει το ανοιχτό αρχείο με αναγνωριστικό αριθμό fileDesc. Σε περίπτωση επιτυχίας επιστρέφει 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση BF_PrintError.

int BF_GetBlockCounter(
int fileDesc /* αναγνωριστικό αρχείου block */)

Η συνάρτηση Get_BlockCounter δέχεται ως όρισμα τον αναγνωριστικό αριθμό fileDesc ενός ανοιχτού αρχείου από block και βρίσκει τον αριθμό των διαθέσιμων block του, τον οποίο και επιστρέφει σε περίπτωση επιτυχούς εκτέλεσης. Σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση BF_PrintError.

int BF_AllocateBlock(
int fileDesc /* αναγνωριστικό αρχείου block */)

Με τη συνάρτηση BF_AllocateBlock δεσμεύεται ένα καινούριο block για το αρχείο με αναγνωριστικό αριθμό fileDesc. Το νέο block αρχικοποιείται με μηδενικά και δεσμεύεται πάντα στο τέλος του αρχείου, οπότε ο αριθμός του block είναι BF_GetBlockCounter(fileDesc) - 1. Σε περίπτωση επιτυχίας επιστρέφει 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση BF_PrintError.

int BF_ReadBlock(
int fileDesc, /* αναγνωριστικό αρχείου block */
int blockNumber, /* ο αριθμός ενός δεσμευμένου block μέσα στο αρχείο */
void** block /* δείκτης στα δεδομένα του block (αναφορά) */)

Η συνάρτηση BF_ReadBlock βρίσκει το block με αριθμό blockNumber του ανοιχτού αρχείου με αναγνωριστικό αριθμό fileDesc. Η μεταβλητή block αποτελεί ένα δείκτη στα δεδομένα του block, το οποίο θα πρέπει να έχει δεσμευτεί. Σε περίπτωση επιτυχίας, επιστρέφει 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση BF_PrintError.

int BF_WriteBlock(
int fileDesc, /* αναγνωριστικό αρχείου block */
int blockNumber /* ο αριθμός ενός δεσμευμένου block μέσα στο αρχείο */
)

Η συνάρτηση BF_WriteBlock γράφει στο δίσκο το δεσμευμένο block με αριθμό blockNumber του ανοιχτού αρχείου με αναγνωριστικό αριθμό fileDesc. Σε περίπτωση επιτυχίας, επιστρέφει 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση BF_PrintError.

void BF_PrintError(char* message /* μήνυμα προς εκτύπωση */)

Η συνάρτηση BF_PrintError βοηθά στην εκτύπωση των σφαλμάτων που δύναται να υπάρξουν με την κλήση συναρτήσεων του επιπέδου αρχείου block. Εκτυπώνεται στο stderr το message ακολουθούμενο από μια περιγραφή του πιο πρόσφατου σφάλματος.

Συναρτήσεις CS (Column-Store File)

Στη συνέχεια περιγράφονται οι συναρτήσεις που καλείστε να υλοποιήσετε στα πλαίσια της εργασίας αυτής και που αφορούν τα αρχεία column-store. Όπου το κρίνετε εφικτό, μπορείτε να χρησιμοποιήσετε και τις συναρτήσεις για αρχεία σωρού που αναπτύξατε στην πρώτη άσκηση.

void CS_Init()

Με τη συνάρτηση CS_Init πραγματοποιείται η αρχικοποίηση του επιπέδου CS.

int CS_CreateFiles(
char **fieldNames, /* τα πεδία της δομής Record */
char *dbDirectory /* φακελος που περιέχει όλα τα αρχεία */)

Η συνάρτηση CS_CreateFiles χρησιμοποιείται για τη δημιουργία και κατάλληλη αρχικοποίηση άδειων αρχείων, ένα για κάθε πεδίο που προσδιορίζεται από τον πίνακα fieldNames (και πρόκειται ουσιαστικά για τα πεδία της δομής Record). Τα αρχεία θα δημιουργούνται στο φάκελο με μονοπάτι dbDirectory. Τα ονόματα των αρχείων θα είναι της μορφής CSFile_όνομα-πεδίου, π.χ. το αρχείο που θα περιέχει names, θα ονομάζεται CSFile_name.

Δεδομένου ότι σε αυτή την περίπτωση θα έχετε να διαχειριστείτε ταυτόχρονα πολλά αρχεία, ειδική πληροφορία εντοπισμού όλων των αρχείων θα αποθηκεύεται σε ένα νέο ειδικό αρχείο με όνομα π.χ. Header_Info. Η πληροφορία θα περιλαμβάνει τα ονόματα των αρχείων που βρίσκονται στο φάκελο και οτιδήποτε εσείς θεωρείτε απαραίτητο. Το αρχείο αυτό είστε ελεύθεροι να το δημιουργήσετε και διαχειριστείτε όπως θέλετε. Μπορείτε π.χ. να κάνετε πάλι χρήση των BF συναρτήσεων, είτε χρήση fopen κλπ.. Σε περίπτωση που εκτελεστεί επιτυχώς η συνάρτηση, επιστρέφεται 0, αλλιώς -1.

int CS_OpenFile(
HeaderInfo* header_info, /* δομή ανοιχτών αρχείων */
char *dbDirectory /* φακελος που περιέχει όλα τα αρχεία */)

Η συνάρτηση CS_OpenFile ανοίγει το αρχείο “header_info” που βρίσκεται στο dbDirectory και επιστρέφει ένα struct που περιέχει τους File Descriptors για όλα τα αρχεία. Η συνάρτηση αυτή πρέπει από τις πληροφορίες που θα βρει στο header_info, να ανοίξει τα εν λόγω αρχεία και να διαβάσει από το πρώτο μπλοκ του καθενός την πληροφορία που αφορά το αρχείο column-store. Αποθηκεύει στο struct HeaderInfo τον αναγνωριστικό αριθμό ανοίγματος κάθε αρχείου, όπως αυτός επιστράφηκε από το επίπεδο διαχείρισης μπλοκ. Αν το αρχείο που ανοίχτηκε δεν πρόκειται για αρχείο column-store, τότε αυτό θεωρείται επίσης περίπτωση σφάλματος. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφεται 0, ενώ σε διαφορετική περίπτωση -1.

int CS_CloseFile(
HeaderInfo* header_info /* δομή ανοιχτών αρχείων */)

Η συνάρτηση CS_CloseFile βρίσκει τα fileDesc των αρχείων που είναι ανοικτά και τα κλείνει. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφεται 0, ενώ σε διαφορετική περίπτωση -1.

int CS_InsertEntry(

HeaderInfo* header_info,	/* δομή ανοιχτών αρχείων */
Record record	/* εγγραφή προς εισαγωγή */)

Η συνάρτηση CS_InsertEntry χρησιμοποιείται για την εισαγωγή μίας εγγραφής στα αρχεία column-store. Η πληροφορία για τα αρχεία CS διαβάζεται από τη δομή HeaderInfo, ενώ η εγγραφή προς εισαγωγή προσδιορίζεται από τη δομή Record. Η τιμή του κάθε πεδίου προστίθεται στο τέλος του αντίστοιχου αρχείου, μετά την τρέχουσα τελευταία τιμή που υπήρχε. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφεται 0, ενώ σε διαφορετική περίπτωση -1.

void CS_GetAllEntries(

HeaderInfo* header_info,	/* δομή ανοιχτών αρχείων */
char *fieldName,	/* όνομα του πεδίου για το οποίο γίνεται ο έλεγχος */
void *value,	/* τιμή του πεδίου προς σύγκριση */
char **fieldNames,	/* ποια πεδία να επιστρέφονται για κάθε value */
int n	/* το πλήθος των πεδίων του πίνακα fieldNames*/)

Η συνάρτηση αυτή χρησιμοποιείται για την εκτύπωση όλων των εγγραφών οι οποίες έχουν τιμή στο πεδίο με όνομα fieldName ίση με value. Οι αναγνωριστικοί αριθμοί ανοίγματος αρχείων δίνονται στη δομή HeaderInfo. Η παράμετρος fieldName μπορεί να πάρει για τιμή κάποιο από τα ονόματα των πεδίων της εγγραφής. Σε περίπτωση που η τιμή του fieldName και του value είναι ίση με NULL, να εκτυπώνονται όλες οι εγγραφές. Να εκτυπώνεται επίσης το πλήθος των μπλοκ που διαβάστηκαν.

Καλείστε να γράψετε και να παραδώσετε κώδικα που να αποδεικνύει την ορθότητα του προγράμματός σας χρησιμοποιώντας τα αρχεία δεδομένων που έχουν δοθεί.

Σχόλια για την Υλοποίηση

Όπως πάντοτε, αναμένεται καλός σχολιασμός του προγράμματος, και εσωτερικός (ανάμεσα στις γραμμές κώδικα) και εξωτερικός (στην αρχή κάθε ρουτίνας). Ένας γενικός κανόνας είναι να σχολιάζετε τα προγράμματά σας σαν να πρόκειται να τα δώσετε σε κάποιον άλλον ο οποίος θα τα επεκτείνει και ο οποίος δεν έχει ιδέα για το τι κάνατε όταν τα γράφατε (και δεν μπορεί ούτε να σας βρει να σας ρωτήσει).
Επίσης, θα πρέπει να ελέγχετε για διάφορα σφάλματα που μπορούν να προκύψουν και να βεβαιωθείτε ότι ο κώδικάς σας τερματίζει ομαλά, με μηνύματα που έχουν νόημα, σε όλες τις εισόδους που ικανοποιούν την παραπάνω περιγραφή.

Λεπτομέρειες εργασίας και παράδοσής της

Η εργασία είναι ομαδική, **2 ή 3 ατόμων**.

Γλώσσα υλοποίησης: C / C++ (Χωρίς χρήση STL και string)

Περιβάλλον υλοποίησης: Θα χρησιμοποιήσετε το BF επίπεδο που σας δίνετε όπως στη προηγούμενη εργασία. Μπορείτε να αναπτύξετε την εργασίας σας σε Linux (gcc 4.3+) ή Windows (Visual Studio

2012) αλλά η εξέταση θα γίνει στα Linux της σχολής συνεπώς τα προγράμματά σας πρέπει να μεταγλωττίζονται και να τρέχουν στα Linux της σχολής.

Παραδοτέα: Τα αρχεία πηγαίου κώδικα (sources) και τα αντίστοιχα αρχεία κεφαλίδας (headers) καθώς και αρχείο readme με περιγραφή / σχόλια πάνω στην υλοποίησή σας. Η παράδοση θα γίνει μέσω της πλατφόρμας e-class.

Προθεσμία Παράδοσης: 25 Ιανουαρίου 2016, 11:59 μμ.