Database Systems Lab

SESSION 6

# Working with Linked Data

In this lab session, you will enhance the ability of PDS to work with linked data. Linked data is also referred to as related data. For example, we can link student data without course data based on the courses the student is enrolled in.

# The following description shows the high-level changes to existing functions and new functions to be added to PDS

struct PDS\_NdxInfo{ // NO CHANGE

int key;

int offset;

int is\_deleted; // 0 - not deleted, 1 - deleted

};

struct PDS\_RepoInfo{

char pds\_name[30];

FILE \*pds\_data\_fp;

FILE \*pds\_linked\_data\_fp; // NEW

FILE \*pds\_link\_fp; // NEW

FILE \*pds\_ndx\_fp;

int repo\_status;

int rec\_size; // For fixed length records for main repo

int linked\_rec\_size; // NEW

struct BST\_Node \*pds\_bst;

int rec\_count;

};

// NEW

// Structure to link parent and child keys

struct PDS\_link\_info{

int parent\_key;

int child\_key;

};

extern struct PDS\_RepoInfo repo\_handle;

// pds\_create - CHANGED

// Create empty repo data file (.dat) and index file (.ndx)

// Create empty linked repo data file (.dat) only. No need for .ndx file

// Create empty link data file (.dat) by concatenating the two repo names

// Example: if repo\_name is "student" and linked\_repo\_name is "course", then link data file should be named student\_course.dat

int pds\_create(char \*repo\_name, char \*linked\_repo\_name);

// pds\_open - CHANGED

// Open the main data file and index file in rb+ mode

// If linked\_repo\_name is NOT NULL

// Open the linked data file in rb+ mode (there is no index file for linked data)

// Open the link file in rb+ mode

// end if

// Update the fields of PDS\_RepoInfo appropriately

// Build BST and store in pds\_bst by reading index entries from the index file

// Close only the index file

int pds\_open( char \*repo\_name, char \*linked\_repo\_name, int rec\_size, int linked\_rec\_size );

// pds\_load\_ndx - NO CHANGE

// Internal function used by pds\_open to read index entries into BST

int pds\_load\_ndx();

// put\_rec\_by\_key - NO CHANGE

// Seek to the end of the data file

// Create an index entry with the current data file location using ftell

// Add index entry to BST using offset returned by ftell

// Write the key at the current data file location

// Write the record after writing the key

int put\_rec\_by\_key( int key, void \*rec );

// put\_linked\_rec\_by\_key - NEW

// Seek to the end of the linked data file

// No need to create index entry

// Write the key at the current data file location

// Write the record after writing the key

int put\_linked\_rec\_by\_key( int key, void \*rec );

// get\_rec\_by\_key - NO CHANGE

// Search for index entry in BST

// Seek to the file location based on offset in index entry

// Read the key at the current file location

// Read the record after reading the key

int get\_rec\_by\_ndx\_key( int key, void \*rec );

// search by non\_ndx key - NO CHANGE

int get\_rec\_by\_non\_ndx\_key(void \*key, void \*rec, int (\*matcher)(void \*rec, void \*key), int \*io\_count);

// get\_linked\_rec\_by\_key - NEW

// Do a linear search of the given key in the linked data file

int get\_linked\_rec\_by\_key( int key, void \*rec );

// delete by ndx\_key - NO CHANGE

// NOTE: When we delete a key, we need to actually delete the linked records too.

// But we will ignore that part for now.

// This function will now delete only from the main data

int delete\_rec\_by\_ndx\_key( int key );

// pds\_link\_rec - NEW

// Create PDS\_link\_info instance based on key1 and key2

// Go to the end of the link file

// Store the PDS\_link\_info record

int pds\_link\_rec (int key1, int key2);

// pds\_get\_linked\_rec

// Go to the beginning of the link file

// Reset result\_set\_size to 0

// Do a linear search of all link\_info records for matching the given parent\_key

// Store the matching linked key in linked\_keys\_result array

// Keep updating the result\_set\_size

int pds\_get\_linked\_rec(int parent\_key, int linked\_keys\_result[], &result\_set\_size);

// pds\_close - CHANGE

// Open the index file in wb mode (write mode, not append mode)

// Unload the BST into the index file by traversing it in PRE-ORDER (overwrite the entire index file)

// Free the BST by call bst\_destroy()

// Close the index file, data file and linked data file

int pds\_close();

# Testing and Demo

1. You will NOT work with Contacts in this LAB!
2. Create YOUR OWN TWO new structs indicating a parent struct and a child struct
3. Write your own NON-INTERACTIVE test program where you call the various PDS functions to demonstrate the functionality on the your own parent-child struct examples

# Submission

Upload the following to LMS as a zipped file (IMTXXXXXXX.zip):

* + - rollno\_pds.c
    - pds.h
    - One additional .h file indicating TWO parent-child struct definitions
    - Your non-interactive testing program
    - bst.h and bst.c