Assignment-1

Instructor: Nitin Awathare

Due Date: Nov 14, 2025 11:59 PM IST

This assignment requires you to implement the stated objective on the toyDB database, for which the necessary files have been provided. The toyDB system is a simplified database that contains the two fundamental lower layers of a database management system (DBMS).

The first is the Paged File (PF) layer, which is responsible for handling storage at the level of individual pages within a file. This layer abstracts away low-level file operations and provides an interface for managing fixed-size data pages efficiently.

The second is the Access Method (AM) layer, which builds upon the PF layer to support B+ tree–based access structures. This layer provides efficient indexing and retrieval operations, enabling structured access to data.

Detailed descriptions of the interfaces exposed by these two layers are available in the provided documentation files (pf.ps and am.ps). In addition, the C source code for both the PF and AM layers has been included, giving you the necessary foundation to understand their implementation and extend the functionality as required for this assignment.

In addition to the database code and documentation, the assignment package also includes test data that students can use to carry out and validate their implementations. This data is bundled in the file data.tar.gz, which, once extracted, provides a collection of databases specifically designed for use with the assignment.

The datasets are stored as simple text files, each representing a table of information. These files contain dummy data that has been carefully prepared solely for the purpose of this assignment. The structure of the data closely resembles realistic academic information, thereby allowing students to practice with data that feels practical while still being simplified for instructional use.

For instance, one of the files, the courses file, contains records of all courses offered across different departments. Each record includes fields such as the course number, the course name, and whether the course is intended for postgraduate (PG) or undergraduate (UG) students. The records are stored in a plain-text format, with individual fields separated by semicolons (;)

Given this, the objectives of this assignment is as follows:

Objective:

- Implement page buffering for the PF layer with support for two page replacement strategies—LRU and MRU (the latter being particularly useful for sequential file access). The replacement strategy should be selectable when opening a file. Each page must maintain a dirty flag to track modifications, with an explicit call provided to mark a page as updated. The buffer pool size should be configurable as a parameter. Additionally, the system must collect and report statistics, including the number of pages accessed (read/write) and both logical and physical I/O counts for different mixtures of read and write queries. You must plot a graph where X-axis represents different mixture of read/write queries while y-axis represents your statistics.

- Build a slotted-page structure on top of the PF layer to store variable-length records, using it to manage student data. The system should support record insertion, deletion, and sequential scanning. Additionally, gather performance metrics such as space utilization and compare it with the static management of the records. Your results should be presented in a table, taking into account different maximum record lengths during static record management.

- The AM layer can be used to build an index on a file. This can be done in two ways:

  1. Starting with a file that already contains records and creating the index in a single operation.
  2. Beginning with an empty file and building the index incrementally through multiple inserts.

  Evaluate both approaches for constructing an index on the *Student* file using the **roll-no** field as the key. If the file is pre-sorted, more efficient bulk-loading techniques for index construction are available

(discussed in the class)—implement one such method and compare its performance (query completion time, number of page accessed) with the two different way of indexing (listed above).

Tips for testing: Before beginning the implementation, you are advised to familiarize yourself with the code hierarchy of the ToyDB database and test it using basic read queries without making any modifications.

SUBMISSION INSTRUCTIONS:

1. The assignment should be done in a group consisting of a maximum of two students.
2. The code should follow programming etiquette with appropriate comments.
3. Add a **README** file which includes a description of the code and gives detailed steps to compile and run the code.
4. Zip all your code as a single file with the name **rollno1-rollno2.tar.gz** and upload it to Google class room. Only one group member should upload the file.
5. Please note that you won't be allowed to make any changes to the code once submitted. You can only make changes to the config file Any changes to the code will result in 20% penalty of total marks. The penalty will increase with the amount of changes done.