PicoDB

Lightweight File-Based Database System

Supervisor:

Dr. Kazi Muheymin-Us-Sakib

Professor

IIT,UNIVERSITY OF DHAKA

Name: Ekramul Hoque

Roll: 1628

Department: IIT

Institution: University Of Dhaka

Introduction to PicoDB

Lightweight & File-Based

PicoDB is a compact, file-based database system developed using C++.

Internal Workings

Designed to provide deep insight into the internal mechanisms of databases.

Custom Implementation

Features custom storage, indexing strategies, and query execution processes.

PicoDB Project Description

Core Functionality

CREATE TABLE: Define new database tables.

INSERT: Add new data records.

EXIT: Terminate the database session cleanly.

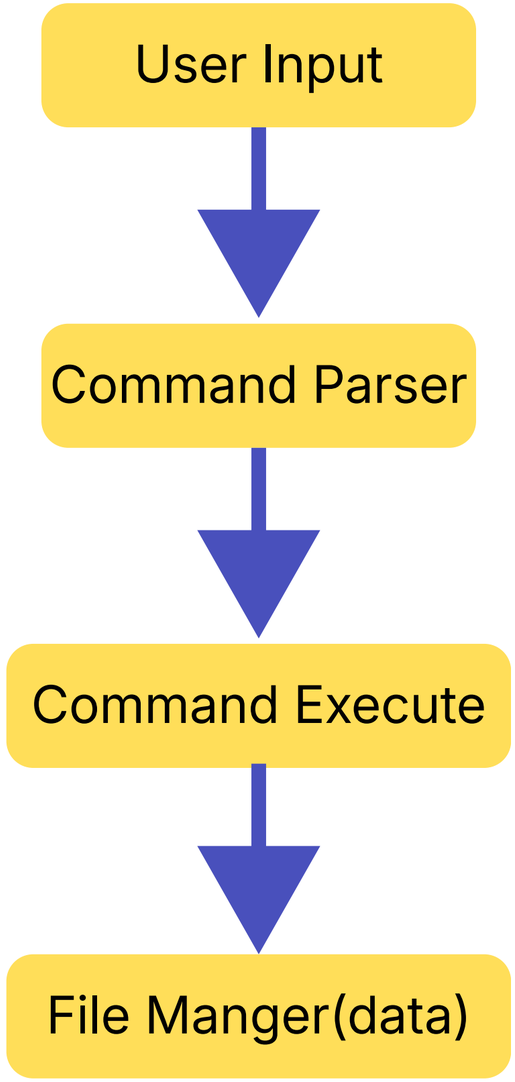
SHOW TABLE: Display table schemas and contents.

SELECT: Retrieve data based on criteria.

System Architecture: Flow Diagram

This high-level view illustrates how user commands are transformed into actions that interact with the underlying storage and indexing layers.

- User input is meticulously parsed into structured commands.
- The Command Executor directs operations to the storage and index layer.
- Data is read directly from disk using these calculated offsets, ensuring efficient retrieval.
- The Index provides precise file offsets for data location.



Data Storage Design

.meta File

Stores table schemas and definitions of column types.

.data File

Contains the actual binary storage of records.

.hashidx File

Dedicated to storing the hash index (current implementation).

.bptidx File

Reserved for the planned B+ Tree index implementation.

Completed Features (Mid-Term Progress)

→ **Table Creation**
Full support for defining tables with specified schemas.

→ **Record Selection**
Efficient retrieval of records based on exact matches.

→ **Table Display**
Functionality to show the structure and contents of tables.

→ **Record Insertion**
Ability to insert new data records into tables.

→ **Hash-Based Indexing**
Implemented a robust hash index for accelerated data access.

→ **Clean Exit**
Graceful handling for exiting the database system.

Key Data Structures & Algorithms Utilized

Data Structures

- **Hash tables:** Fundamental for efficient indexing and quick data lookups.
- **Vectors:** Employed for dynamic storage management of various components.
- **Binary file streams:** Crucial for direct, low-level interaction with data files on disk.
- **Custom record encoding:** Tailored serialization format for data records.

Algorithms & Techniques

- **Hash-based lookup:** Fast retrieval mechanism based on hash values.
- **Varint encoding:** Space-efficient encoding for variable-length integers.
- **File offset addressing:** Precise direct access to data blocks within files.
- **Binary serialization & deserialization:** Processes for converting data to/from binary format for storage and retrieval.

Motivation

Internal Mechanics
Gain a profound understanding of how real-world databases function from the inside out.

Beyond Theory
Translate theoretical knowledge of indexing into practical, hands-on experience.

Hands-On Experience
Acquire practical skills in file I/O, index structures, query execution, and low-level data representation.

Challenges

Variable-Length Records
Designing and managing records with dynamic sizes in binary files.

Index-Data Consistency
Ensuring the hash index accurately reflects the state of the data file.

File Offset Handling
Safely and correctly managing file pointers and offsets for data access.

Future Work (Post Mid-Term)

Planned Core Features

- **B+ Tree Indexing:** Implement for efficient range queries and optimized data retrieval.
- **UPDATE Operation:** Enable modification of existing data records within tables.
- **DELETE Operation:** Introduce functionality for removing specific records.
- **Index Persistence:** Improve the durability and reliability of indexing structures across sessions.

Advanced Extension (Future)

- **Socket Programming:** Integrate network capabilities to facilitate remote interaction.
- **Multi-Client Access:** Allow multiple users to connect and interact with PicoDB over a local network.

THANK YOU