Written by Archer Heffern
For System Creation

Database in C

Description

A database is an organized collection of structured data.
A database management system is a program which manages the database.

The data should be stored in persistent memory (Files)

The basic operations of a database are *(Wikipedia)*

- **Data definition** – Creation, modification and removal of definitions that define the organization of the data.
- **Update** – Insertion, modification, and deletion of the actual data
- **Retrieval** – Providing information in a form directly usable or for further processing by other applications. The retrieved data may be made available in a form basically the same as it is stored in the database or in a new form obtained by altering or combining existing data from the database.

There are many ways to structure data in a database, just think back to Data Structures! For this project we will use tabular data - think Excel. You can have multiple tables (data types), create tables (data definition, and create/read/update/delete an entry from a table.

Goal

Create a tabular data database management system with the following features:

1. CLI interface
2. Basic Query Language for CRUD Operations and Data Type Creation
3. Meta Commands .exit and .help
4. Persistent storage as a single binary file

CLI Interface

For now, we will be able to interface with the database through the command line. Later, we will transition this to a network interface. Either way, the database itself will take strings (metacommands and query language) as input. Decouple the IO logic from the actual database logic so this transition later is super easy.

Query Language and Meta Commands

Similar to the Shell project, we can either type commands, or builtins - in this context called Query Language and metacommands respectively.

Written by Archer Heffern
For System Creation

Query Language**:**The Query Language specification is up to you, although you may want to take inspiration from SQL - The Query Language usually used for Relational Databases. You will want to implement syntax to create data types and perform CRUD operations.

Data types created should allow fields to have 2 data types - string and int.
eg. CREATE TABLE User (name string, age int, height int)
Again, use whatever syntax you want, as long as it makes some sense.

Meta Commands**:** Implement at least 2 metacommands .quit and .help.

Persistent Storage in Binary Format

All data types, data, and metadata should be stored in a single file as binary.

Parts

This sounds very complex, so let's break this down into smaller steps. Ask for a code review before moving onto a next step. If you want to do this another way, feel free to.

MVP

- Single data type (No data type creation)
- CRUD operations
- no persistence
- basically just an array
- Command line UI
- Meta Commands .exit and .help

Example

Plain Text Persistence

Objective: Serialization and Deserialization of Data

I would suggest doing this before replacing it with binary format, since it will be much easier to debug.

- Should Update all CRUD operations to either store or read from the database.
- Since there is only 1 data type, this should be very simple. You will need to think of how data is delimited.
- Remember, this project calls for only 1 file

When updating and deleting, you should not be rewriting any data except for the data we want to modify

Written by Archer Heffern
For System Creation

Add Multiple data types (Hardcoded)

Objective: Designing More Interesting File Formats using headers

Add 1-2 more data types to your program with different sizes from your original struct, allowing for Persistent CRUD operations of more than 1 datatype. Eg. Trees, Users, and Frogs.

Notes:

1. You do not need a new Create, Read, Update, and Delete function for each data type
2. This should all be done with a single database file.

Hint: Encoded data should have a header or some feature to indicate its type

Allow for creation of data types

Objective: Designing more interesting File Formats using recursive headers

Implement a feature to create new data types (Tables). Data Type Fields (Fields) should support 2 data types, string and integer.

Example Syntax. CREATE USER (name str, age int, height: int)

Again, this should all be done with a single database file

Hint: We will need a section in our database file for storing our schemas (data type declarations)

Binary Format Persistence

Same idea as plain text, except now in binary format. Hopefully if you designed your project well, this should just require implementing some methods and hot swapping in some places.

Submission

Create a git repo to track your project, push it to github, and send the git repository URL to me.

Below is the list of files in the git repository:
db.c : The database management system
.git : The git repository
.gitignore : Should list all executables
start.sh or makefile : For compiling project
README.md : A readme file containing instructions for compiling, running, and any other information
Include any other helper files used in your project

Written by Archer Heffern
For System Creation

Additional Ideas for Optimizations and other Improvements

1. Better Select Function
    a. Needs improvement of select function and creation of a query language
2. Enable modeling of relationships between data types (Use pointers)
    a. Requires extension of language to allow for querying this way
3. Speed Optimization: Store some data in program memory for lookup speed, currently we always check disk for data, which is very slow.

Advanced Improvements:

4. Multiple users and managing of permissions and accounts
5. Durability: (Retains memory through computer failures) with WAL (write ahead log)
6. Better Data Structures for faster lookup (B-Trees)

Good Software Features

7. IPC Interface eg. Sockets (users can connect from other processes (and possibly other computers))
8. Daemonization: Always running, executes on system bootup, and can be managed with the init system
9. Versioning: Software changes, so ensure either backwards compatibility or a section in your database header for the version of the database format

Data: <TODO>

To translate data from one form to another, you will need to create a serializer, and deserializer. Example: Converting an Array to CSV.

Sizing/framing
serializers and deserializers

Data Design:

- Headers / Metadata
- Encapsulation
    - Preknown size assigned with data type
    - Header containing data type size
    - Delimiter

https://www.sqlite.org/fileformat.html