Assignment II: DB Machine

In this second assignment, you will extend the libchidb library with the χdb Database Machine, a virtual machine specifically designed to operate on χdb files that includes instructions such as “Create a new table”, “Find a record with key k”, etc. This assignment is divided into the following steps:

1. Database Machine
2. Schema loading
3. Simple code generation
4. API functions

The implementation of these three steps is actually not too hard, and each step can be easily subdivided into smaller tasks that can be distributed amongst multiple developers; the main difficulty of this assignment is that, unlike Assignment I, you are given almost complete leeway in the design of your solution. Although you are provided with the specifications for the database machine instructions and the API of the χdb library, you are not provided with function signatures, structure definitions, etc. that you have to conform to. That is left entirely up to you (and you should give a lot of thought to the design of the solution before a single line of code is written).

Before you get started

Before you start working on this assignment, make sure you read The χdb Architecture. You will also need a complete implementation of Assignment I.

Step 1: Database Machine

Implement a χdb Database Machine as specified in “The χdb Architecture”. Your Database Machine must use the B-Tree implementation from Assignment I as its backend.

Step 2: Schema loading

Write code that, given a valid χdb file, will load its schema into memory. We suggest that you store this information in the chidb struct. Remember that the database schema is contained in a B-Tree and that the schema for each individual table or index is stored as the SQL statement used to create the table or index. You will be provided with a SQL parser that, given a string containing a SQL statement, produces an in-memory representation of the SQL statement that is easy to manipulate programmatically.

Note that, in a first cut, this code will have to use low-level B-Tree functions. However, if you implement all steps, the schema loading can be reimplemented more elegantly as just sending off a query to the special schema table.
Step 3: Simple code generation

Write a code generator that, based on the in-memory representation generated by the SQL parser, produces code for the χdb Database Machine. At this point, you only need to generate code for the following statements:

1. SELECT a list of columns (i.e., you don’t need to support SELECT *) from a single table and with, at most, one condition in the WHERE clause. The condition will be of the form field op value.

2. INSERT statements.

Step 4: API functions

Implement all the API functions specified in “The χdb Architecture” in main.c. Note that, although this file includes an implementation of chidb_open and chidb_close, these two functions still need additional work (a barebones implementation is provided since the automated tests rely on them).