CPSC 531 - Advanced Databases - Fall 2019

Project 3, due December 12

RDBMS and Database

The RDBMS for this project is one of the following programs that you and your team will implement:

1. A program in C or C++ using the GNU dbm library for indexing.
2. A program in Python using the `dbm` or `shelve` modules for indexing.
3. A program in Java using Oracle Berkeley DB Java Edition for indexing.

The database format is a binary file of disk blocks. The disk block size is 4,096 bytes, and the blocking factor bfr is 10. Each record is the equivalent of the following SQL DDL statement:

```
CREATE TABLE Person
(
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20) NOT NULL,
    job VARCHAR(70) NOT NULL,
    company VARCHAR(40) NOT NULL,
    address VARCHAR(80) NOT NULL,
    phone VARCHAR(25) NOT NULL,
    birthdate DATE NOT NULL,
    ssn VARCHAR(12) NOT NULL,
    username VARCHAR(25) NOT NULL,
    email VARCHAR(50) NOT NULL,
    url VARCHAR(50) NOT NULL
);
```

Strings are composed of ASCII characters and are null-terminated. Dates are stored as three 32-bit integers in native byte order representing the day, month, and year.

There are two test databases: `small.bin.gz`, of size 40,960 bytes, containing 100 records, and `large.bin.gz`, of size 4 GiB, containing over 10 million records. These files are compressed with GNU GZip for download, and should be uncompressed before use.

Indexes will be created as DBM files using one of the libraries listed above.

Platform

You may use any platform to develop and test your code, but note that per the Syllabus the test environment for projects in this course is a Tuffix 2019 Edition r2 Virtual Machine.

Libraries

The Python 3 standard library included with Tuffix includes all necessary modules.

For C or C++ you may use any ANSI standard version supported by the GCC or Clang compilers included with Tuffix . The GNU dbm library is already installed and can be included by linking with the `-lgdbm` switch, but the `gdbm.h` header files will need to be installed with the following command:

```
$ sudo apt install --yes libgdbm-dev
```

Tuffix includes OpenJDK 11. You will need to create an Oracle account in order to download Berkeley DB Java Edition 7.5.11. You may use either the Base API (`com.sleepycat.je`) or the Collections (`com.sleepycat.collections`).

Reading binary files

You may use any method to read binary files, but you may find the following useful:

- C: `fread()` into a `struct`.
- C++: `std::basic_ifstream::read` into a `struct` via `reinterpret_cast`.
- Python: `read()` into a `bytes` object, then decode with the `struct` module.
- Java: `ByteArrayInputStream.read()` or `FileChannel.read()` into a `ByteBuffer` or `String`.

Queries

Each of the following queries should be implemented as separate programs. For queries that use an index, write two separate programs - one to build the index, and one to use the index to run the query.

In each case, test your program using `small.bin` first to verify that it works correctly before attempting the query on `large.bin`.

Use the UNIX `time` command to measure how long each query takes, and include the results in your submission.

Tip: based on the time you measure for `small.bin`, you may want to do a back-of-the-envelope estimate before starting queries on `large.bin`.

Query 1 - Table scan

Read the file block-by-block, list the SSN, first name, and last name of all users under age 21.

Query 2 - Uniqueness check

The SSN is supposed to be a unique identifier, but it was not declared `UNIQUE` above. Read the file block-by-block, using a DBM database to check whether the SSN has been seen before. Report any duplicates.

Query 3 - Secondary index

Use a DBM database to create a secondary index on birthdate, then loop through all items in the index to find the location on disk of all users under age 21. Read only the relevant disk blocks in order to list the SSN, first name, and last name of all users under age 21.

Query 4 - Clustered index

Create a clustered index on birthdate by sorting the data file and creating sparse DBM index entries for each disk block. Use this index to repeat the previous query.

Submission

Submit your project by uploading your source code, build files or documentation, and any other relevant artifacts to the `project3/` subdirectory of the folder that was shared with you on Dropbox. Do **not** include data files, particularly copies of `large.bin`, compressed or not. See below for penalties if you do so.

You may work alone, or make a single submission for a team of 2-3 students. If you work in a team, only one submission is required, but for safety consider uploading copies to each team member's submission folder. (Make certain, however, that the copies are the same in each case; the instructor will not attempt to ascertain which is the “real” submission.)

A printed submission sheet will be provided on the due date. To finalize your submission, fill out the sheet with the requested information and hand it in to the professor by the end of class. Failure to follow any submission instructions exactly will incur a **10%** penalty on the project grade for all team members.