15-415 Databases
Project #2 (Index Library)

Assigned:	Thursday, October 6, 2011
Proposal:	Thursday, October 13, 2011
Key indexing or Datastore:	Thursday, October 20, 2011
Record Indexing/Caching/Buffering:	Thursday, October 27, 2011
Final Deadline:	Thursday, November 3, 2011

Objective

This assignment asks you to design and implement an API to construct and maintain an index and a data-store for structured data. Application programmers should be able to use your API to conveniently and easily index their data and subsequently insert, find, and remove items while maintaining the index. It should support both exact-match and range queries.

Furthermore, your library should be able to support multiple indexes, multiple indexes of the same (or different) data based on different keys, and currency both within indexes and among indexes. Keys should be composable from any tuple of fields. The index should be able to enforce primary key constraints, if requested by trhe application programmer. Needless to say, your library should make use of features of B+ and B* trees.

Teams

Teams of up to three people are permitted, with the caveat that each team member must (a) do equal portions of the coding and design. This may be assessed by one-on-one or team interviews and code reviews for grading purposes. Be prepared.

Technologies and Design Details

We are taking a very broad view of this project. You're API can be a traditional UNIX local library. It can be a traditional pair of a server program and a client library to support a network database. It can be a Web Service, or a client library and a Web service. It must support use, locally or remotely, by an application written using tools available on the Andrew UNIX systems. It may be written in any available language and support any available language.

We are also giving you space to design your system. You may decide which of the B-tree family features you'll use, how you'll do your locking, how you'll format your data blocks, etc.

Performance

We do expect that you will have some performance-enhancing features, such as buffering and caching, redistributions of keys and merging of pages, etc. Please be sure to document these as part of your proposal and final deliverables as discussed below (Proposal; Final Deadline and Deliverables). Think carefully about what you are promising about when data is stable, and document this so the application programmer can plan accordingly. You might want to include flush options, relaxed options, etc, in your design.

Input Data Format, Data Store

Your API should provide a function capable of bulk-loading a CSV file into an indexable data store organized any way you'd like (think leaf-level blocks). The data store should be indexable by more than one index based on more than one key field.

In addition to supplying the path to the CSV file, the caller should specify, at least, the field to use as the key for the index. This field should be specified by field number, with the left most field in the CSV file being numbered *0*.

Indexes and Indexing

Your API should provide a function capable of building an index from an existing data store created as described above (Input Data Format, Data Store). It should allow the user to specify the key to be indexed. A key should be composable from any tuple of fields. If requested, the API should be able to enforce primary key constraints, e.g. fail and notify upon encountering a duplicate primary key. The user should be able to create multiple indexes for the same data store to index by different keys.

The system should also be able to insert, remove, and find records by key. It should be able to perform these operations both by exact-match and range queries.

Sample Application

I order to ensure that we can test your API, without doing a huge amount of programming ourselves, you should include a sample application. It does not need to be interactive. But, it does need to demonstrate to us that all of your features work. Please ensure that it demonstrates multiple indexes and concurrency.

Proposal

By 11:59PM Thursday, October 13th, you should electronically submit a proposal containing three documents:

(a) API documentation. This should explain to an application programmer how they'll use your API. You might choose to model it on UNIX "man pages" or Java's API documentation, for example.

(b) Design document. This should explain to us how you intend to structure your data store, the features of your index tree, and how you'll handle concurrency, buffering and caching, and the stability of data.

(c) Detailed timeline: Who will do what and when to ensure a successful project?

Key Indexing or Datastore

By Thursday, October 20th, 2011, you should electronically submit a checkpoint to us, demonstrating your current progress. We expect that, at the least, by this point in time, you should have either (a) implemented your tree structure and be able to index keys, or (b) implemented the feature to create your datastore from a CSV file.

Record Indexing, Caching, Buffering

By Thursday, October 27th, 2011, you should electronically submit a checkpoint to us, demonstrating your current progress. We expect that, at the least, by this point in time, you should have completed both your index and your data store and should be able to use your index to locate, find, insert, and delete records, even if you cannot yet handle multiple indexes, multiple datastores, or concurrency. We also expect that you'll have most of your caching and or buffering implemented, even if, for example, concurrency control is not yet complete.

Final Deadline and Deliverables

By, 11:59pm on Thursday, November 3, 2011, you should electronically submit the following:

(a) An exact copy of your original proposal
(b) A update document describing any changes you've made since your original proposal
(c) A current copy of your API documentation
(d) Your API implementation
(e) Your sample application, and sample data with which to test it. Depending upon your implementation, this might also include query files
(f) Documentation about how to use your test program.