EECS277 – DATABASE SYSTEMS IMPLEMENTATION

Project 5: Complete Database Server
Due date: May 14 (during the final exam)

This project requires the complete implementation of the database server. Everything has to work correctly and be accessible through the SQL language. Since all the functionality is supposed to have been implemented by this time, the only work that has to be done is to expose the missing parts through SQL commands. There is only one missing command:

- `CREATE INDEX` *`index-name`* `TABLE` *`table-name`* `ON` *`attribute-name`*. This command creates an index *index-name*, i.e., B+-tree, on attribute *attribute-name* from table *table-name*. The index properties, i.e., name, table, attribute, and the name/location of the index file, have to be stored in the catalog and be used in query processing.

  The command iterates over the pages in the heap file corresponding to *table-name* and, for each record, extracts the value of the attribute used to build the index on, together with the page number in the data file and the record index in the page. The pair (page-number, record-index) form the address in the data file. The B+-tree is built by inserting each key in the index data structure. It is valid to assume that enough memory is available to store the entire index in memory both at build time and query time. Once the index is built, it has to be materialized in an index file.

  The output of the command is an index file that stores the B+-tree. Classes `File` and `Page` can be used for this purpose. However, you have to be able to distinguish between an internal page and a leaf page since the records they store have different schema. The schema of a record in an internal page is (index-key, child-page-number). The schema of a record in a leaf is (index-key, data-page-number, record-number). The header of the page has to contain additional metadata, such as the type of the page, the parent page, the left sibling, and the right sibling. Alternatively, a different file format that you design can be used to store the index.

  In order to use an index during query processing, you have to implement a new scan operator `ScanIndex`. You can think of this operator as a combination of `Scan` and `Select`. `ScanIndex` is implemented by searching the index and retrieving only the records satisfying the predicate from the data file. For this, the index can be read completely in memory from the file it was stored in after the construction. Once in memory, the index can be traversed to retrieve the address (page-number, record-index) of all the tuples that satisfy the selection condition. The addresses can be stored in a temporary buffer such as a `List`. In the `GetNext` method from `ScanIndex`, an iterator over the temporary buffer extracts an address that is used to access the corresponding record from the data (heap) file associated with the table. The extracted record is passed to the parent operator.

  `ScanIndex` is used in a query plan whenever the selection predicate corresponding to a table contains conditions involving a single attribute and there is an index on that attribute. You have to support both point queries and range queries over the index.

Implementation

You are required to extend the SQL compiler provided with the project code such that it can handle the new command. This requires modifications to the `yacc` specification and to the compiler data structures. The execution of this command has to be triggered accordingly. You need to add the new `ScanIndex` relational operator and implement it. You also need to extend the Query Compiler you wrote in Project 2 to include `ScanIndex` in query plans whenever deemed appropriate.

Requirements

You have to implement a command-line interface to the database server, similar to `sqlite`. When you start the application, the user is presented with a prompt where SQL commands can be entered. Any SQL statement supported by the compiler has to be executed and the result printed to the corresponding output file. When the execution ends, the command is given back to the user. In addition to the SQL commands, a series of additional commands are useful. `exit` or `quit` ends the application. `schema` prints the schema of all the tables in the catalog. `index` prints the indexes from the catalog. You can think of a few others.

The evaluation consists in executing a series of commands at the prompt. All the queries given to you in previous stages are supposed to work. In addition to these queries, the application is supposed to execute any other syntactically correct SQL statement. The command to create indexes is also supposed to work for the tables in `TPC-H` as well as having the indexes used in queries.