Command Line Interface (CLI) Interpreter for Project4

Introduction

The Command Line Interface (CLI) takes a SQL-like command and executes that command. The CLI is laid on top of the Relation Manager (RM) and Query Engine (QE) layers in Project 4 so that each command is executed via the public methods of the RM and/or QE layers.

Supported Commands

The interpreter supports the following commands.

```
create catalog
create table <tableName> (col1 = <type>, col2 = <type>, ...)
    <type> = int | real | <varchar>
    <varchar> = "varchar(" <length> ")"

create index <columnName> on <tableName>

drop catalog
drop table <tableName>
drop index <indexName> on <tableName>
drop attribute <attributeName> from <tableName>

insert into <tableName> tuple(col1 = <value1>, col2 = <value2>, ...)

SELECT <query>
<query> =
    PROJECT <query> GET "[" <attrs> "]"
    FILTER <query> WHERE <attr> <op> <value>
    INLJOIN <query>, <query> WHERE <attr> <op> <attr>
    IDXSCAN <query> <attr> <op> <value>
    TBLSCAN <query>
    <tableName>

<agg-op> = MIN | MAX | SUM | AVG | COUNT
<op> = < | > | = | != | >= | <= | NOOP
<attrs> = <attr> { "," <attr> }
<numPages> = is a number bigger than 0

print <tableName>
print attributes <tableName>
print index <attributeName> on <tableName>

load <tableName> <fileName>: loads the file which is in data/ folder.
help <commandName>
help
quit | exit
```

Note: All commands are case-insensitive. However, all user defined strings such as table names, attribute names etc. are case-sensitive.

Command Examples

When you first execute CLI, you need to create the system catalog using the following command. Once the catalog is created, you don't need to execute this command again.

```
create catalog
```

You can create a table and insert a record using the following commands:

```
create table Employee EmpName = varchar(30), Age = int, Height = real, Salary = int

insert into Employee tuple(EmpName = sky, Age = 22, Height = 6.1, Salary = 13291)
```

Also, CLI supports a load command that allows users to load records in a file to an existing table. The following screenshot shows how to load records from a file, employee_5, to a table, tbl_employee, and print all records in the table. Furthermore, it also shows how to use SELECT command with FILTER iterator.

```
>>> create table tbl_employee EmpName = varchar(30), Age = int, Height = real, Salary = int
>>> load tbl_employee employee_5
>>> print tbl_employee
EmpName           |  Age  |  Height    |  Salary  |
=================================================
Anettea Belote    |  67   |  6.400000  |  75000   |
Zina Legleiter    |  45   |  6.300000  |  150000  |
Rena Broadus      |  68   |  5.900000  |  250000  |
Lorriane Shimmin  |  49   |  6.600000  |  400000  |
Elvira Binns      |  36   |  5.600000  |  200000  |
>>> SELECT FILTER tbl_employee WHERE Age = 45
tbl_employee.EmpName  |  tbl_employee.Age  |  tbl_employee.Height  |  tbl_employee.Salary  |
=============================================================================================
Zina Legleiter        |  45                |  6.300000             |  150000               |
```

Note that there is a set of example programs under codebase/cli/ directory, which are used for example programs, such as "cli_example_01.cc". Each of the 8 example programs executes a set of CLI commands. These example cases provide a more comprehensive usage of CLI commands. These examples show that (despite syntax shown above, which doesn't show parentheses) that nested queries have to be in parentheses, e.g.,

```
SELECT INLJOIN (INLJOIN employee, salary WHERE Salary = Salary), ages WHERE Age = Age

SELECT PROJECT (FILTER (PROJECT tbl_employee GET [ EmpName, Age ]) WHERE Age >= 45) GET [
EmpName ]
```

Getting and Using Command Line Interface

The codebase has the following source tree.

```
codebase/
├── cli
│   ├── cli.cc
│   ├── cli.h
│   ├── cli_example_XX.cc
│   ├── makefile
│   └── start.cc
├── data
│   ├── ages_90
│   ├── employee_5
│   ├── employee_50
│   └── salary_5
├── ix
│   ├── ix.cc
│   ├── ix.h
│   ├── ixtestX.cc
│   ├── ixtest_extra_X.cc
│   ├── ixtest_util.h
│   └── makefile
├── makefile.inc
├── qe
│   ├── makefile
│   ├── qe.cc
│   ├── qetestX.cc
│   ├── qetest_util.h
│   └── qe.h
├── rbf
│   ├── makefile
│   ├── pfm.cc
│   ├── pfm.h
│   ├── rbfm.cc
│   ├── rbfm.h
│   └── rbftestX.cc
├── readme.txt
├── rm
│   ├── makefile
│   ├── rm.cc
│   ├── rm.h
│   ├── rmtest_XX.cc
│   ├── rmtest_create_tables.cc
│   ├── rmtest_delete_tables.cc
```

```
|   ├── rmtest_extra_X.cc
|   └── test_util.h
└── shared.h
```

Note that the folder, codebase/data/ has csv files that contain a set of records, which can be used for your own checking purpose. Also, the data folder is a folder that CLI program searches for files specified in "load" commands.

Go to the cli folder and type the following commands

```
make clean; make; ./start
```

If you see the following outputs (the folder hierarchy will be different), you are good to go to the next step.

```
************************
SecSQL CLI started
Enjoy!
>>>
```

Now, put all files that you implemented(and will implement in Project4) to the corresponding folders in the above source tree. Then, if you type again the same command in the cli folder, you can start playing by typing "**help**".