SimpleRA

A laughably-minimalist, integer-only Relational Database Management System that makes the author question why they ever bothered to write it up!

Data Systems, Monsoon 2021

17th August, 2021
Tuesday

Important Features

- Relational Algebra Operators
- Integers Only
- No nested queries
- No transaction management
- Single thread programming only
- No identifiers should have spaces in them

Commands

There are 2 kinds of commands in this database.

- Assignment statements
- Non-assignment statements

Note: Not all operators have been implemented, some have been omitted for you to implement in later phases

Non-Assignment Statements

Non-assignment statements do create a new table (except load which just loads an existing table) in the process

- LOAD
- LIST
- PRINT
- RENAME

Non-assignment Statements

- EXPORT
- CLEAR
- QUIT

LOAD

Syntax:

```
LOAD <table_name>
```

- To successfully load a table, there should be a csv file names <table_name>.csv consisiting of comma-seperated integers in the data folder
- None of the columns in the data file should have the same name
- every cell in the table should have a value

Run: `LOAD A`

LIST TABLES

Syntax

```
LIST TABLES
```

This command lists all tables that have been loaded or created using assignment statements

Run: `LIST TABLES`
Run: `LOAD B`, `LIST TABLES`

PRINT

Syntax

```
PRINT <table_name>
```

- Displays the first PRINT_COUNT (global variable) rows of the table.
- Less number of rows can be printed if the table has only a few rows

Run: `PRINT B`

EXPORT

Syntax

```
EXPORT <table_name>
```

- All changes made and new tables created, exist only within the system and will be deleted once execution ends (temp file)
- To keep changes made (RENAME and new tables), you have to export the table (data)

Run: `EXPORT B`

QUIT

Syntax

```
QUIT
```

- Clear all tables present in the system (***WITHOUT EXPORTING THEM***) (temp file - empty)

Run: `QUIT`

Assignment Statements

- All assignment statements lead to the creation of a new table.
- Every statement is of the form

  ```
  <new_table_name> <- <assignment_statement>
  ```

- Naturally in all cases, <new_table_name> shouldn't already exist in the system

Assignment Statements

- CROSS
- PROJECTION
- SELECTION

The following haven't been implemented

- JOIN
- SORT
- GROUP_BY

SELECTION

Syntax

```
<new_table_name> <- SELECT <condition> FROM <table_name>
```

Where <condition> is of either form

```
<first_column_name> <bin_op> <second_column_name>
<first_column_name> <bin_op> <int_literal>
```

Where <bin_op> can be any operator among {>, <, >=, <=, =>, =<, ==, !=}

- The selection command only takes one condition at a time

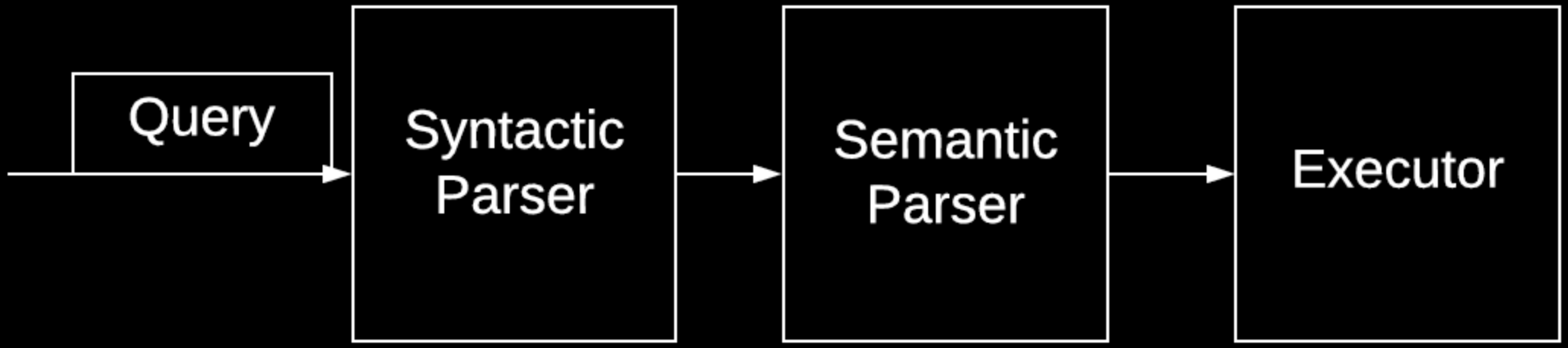
Run: `R <- SELECT a >= 1 FROM A`

`S <- SELECT a > b FROM A`

Internals

- Buffer Manager
- Cursors
- Tables
- Executors

Command Execution Flow



Run: `LOAD A` with debugger

see: load.cpp

Syntactic Parser

- Splits the query into query units

see: syntacticParser.h syntacticParser.cpp

Semantic Parser

- Makes sure your query makes semantic sense

see: semanticParser.h semanticParser.cpp

Executors

Every command(COMMAND) has a file in the executors directory, within that directory you'll find 3 functions

```
syntacticParseCOMMAND
semanticParseCOMMAND
executeCOMMAND
```

Buffer Manager

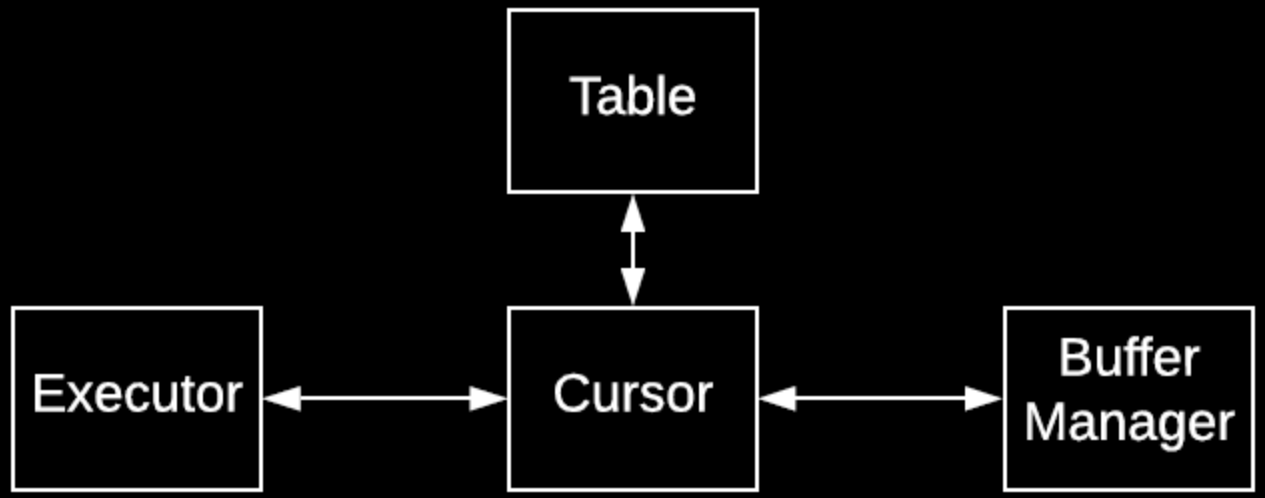
- Load splits and stores the table into blocks. For this we utilise the Buffer Manager
- Buffer Manager follows a FIFO paradigm. Essentially a queue

Table Catalogue

- The table catalogue is an index of tables currently loaded into the system

Cursors

A cursor is an object that acts like a pointer in a table. To read from a table, you need to declare a cursor.



Run: `R <- SELECT a == 1 FROM A` with debugger

Logger

Every function call is logged in file names "log"

Project*

- Phase 1: Code Familiarity
- Phase 2: 2 Phase Merge Sort
- Phase 3: Indexing
- Phase 4: Optimization

* Tentative

Project Rules

- **Plagiarism: F**
- Not sticking to submission guidelines will lead to penalties and at times to scoring 0
- Project phases build on top of each other, failing to do one phase may hinder the rest
- If for any reason you fail to complete the project on time, please mail the Prof directly for extensions and not the TAs, the TAs have no jurisdiction in these cases

Adminstrative Rules

- Moodle is the preferred platform. Informal contact is discouraged, do so at your own expense.
- A "doubts document" will be put up on Moodle.
- If you need to contact the TAs for matters that don't concern the whole class you may mail us here - datasystems_ta_m21@IIITAPhyd.onmicrosoft.com

References

- GitHub Repo - SimpleRA
- Build and run instructions will be provided later