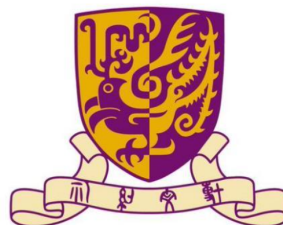


CSC3170 Project Report

121090003 Jingzhi Bao,
120090792 Zimeng Huang, 120090108 Wei Zhou,
120040061 Ningyuan Wang, 119010306 Ruiyi Wang,
119020401 Jiaming Wang, 120090784 Qihua Yin

December 2022



The Chinese University of Hong Kong, Shenzhen

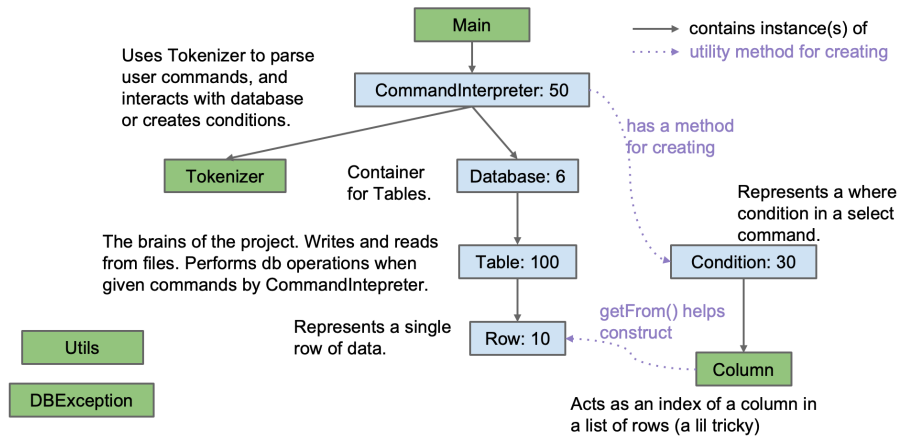
1 Introduction

The communication with the database system using an artificial notation usually known as a *language*, although it is much simpler than any human language. The definition and processing of such languages is an important skill for any computer scientists. We normally think of programming languages such as Java, but there are many other contexts where small, domain-specific languages (DSLs) are appropriate engineering solutions to a design or implementation problem.

Thus, in this way, this project tries to implement a miniature relational database management system (DBMS) that stores tables of data, where a table consists of some number of labeled columns of information. Our system will include a very simple query language for extracting information from these tables.

2 Design

Project Organization (full picture)



Our project mainly focuses on the *Command Interpreter* module and database structures of database system. It will basically support these functions:

1. Interpret SQL language.
2. Support the basic structures of database system (e.g. database files, tables, rows, columns).
3. Do fundamental queries on database system, including *select*, *from*, *where*, *insert into*, *create table* clauses.

The main function is in a public class called `db61b.Main`, which is the most direct way to invoke all the functions. The first line of output of our program identifies the program, and of course it may contain anything. Before reading the first command and on reading each subsequent end-of-line or comment, our program will print the prompt `'> '` (which is followed by a blank as well), or `'...'` to indicate a continuation. The output standards can be referred in exactly the format shown in our results, and when encounters an error, the warning messages will be split on separate lines, starting with the word *error*. In addition, the program will exit without further output when it encounters a quit (or exit) command or an end-of-file in the input. When printing out the contents of a table, the final version of outputs may not follow the order of rows, but it will be in the order specified in the `.db` file or in the columns of the select command from which the table came. Besides, we also provide a *makefile* which is set up to compile everything on the command *gmake* and to run all the tests on the command *gmake* check.

3 Implementation

3.1 Command Interpreter

With the help of the given class **Tokenizer**, we implement the *CommandInterpreter()*, where we can break up all the inputs of the users and turn them into command that our program can execute.

We mainly have 7 kinds of commands here, which are: *Create*, *Load*, *Exit*, *Insert*, *Print*, *Store*, and *Select*.

Create: When command “create table” is found, read the table name, create an object of Table class and name it after the inputted name.

Load: Use *database.put()* to load the data into a existed table.

Exit: Simply exit a statement.

Insert: When “insert into values” is found, read the actual values that the users want to insert next. Create a Row object to catch these values, and insert the new row into a table.

Print: Print the data in a specific Table object.

Store: Use *table.writeTable()* to store the table as a database(.db) file.

Select: When “select” is found, execute *SelectClause()*, which works as follows:

1. Read the column names that the users want to select, until “from” is found.
2. Read the table names that the users want to select the columns from, until “where” is found.
3. Join all the tables.
4. Read all the conditions and use the *ConditionClause()* to validate them.
5. Select the columns in the joined table with the conditions.

3.2 Database

Database is the class that handles table instances. Specifically, the database class supports operations to store, retrieve, and delete tables. We utilize two ArrayList to store table instances and table names.

3.3 Table

We consider *Column* to be an important attribute in the *Table*, and it is only constructed when the database loads '.db' files or executes 'create' clause (for convenience, we call these tables *database tables*, which are stored directory in heap). Namely, the *Column* is constructed at the same time as the 'initial' table.

Idea: the columns of a new table obtained by the 'select clause' is a subset of the 'database tables'. (We define a 'temporary table' is a table created by a 'select clause', otherwise 'standard table'. More specifically, the temporary table is one without explicit name)

To maintain this property, the Table should support two ways of construction:

1. Table(String[] columnTitles), used for *standard table*. (The columns are initialized by string array)
2. Table(List<Column> columns), used for *temporary table*. (The table borrows columns from the table in database)

By emphasizing the importance of *Column*, the role of '_columnTitles' (type: String[]) is diminished. We will just use them as column titles to display to the user.

3.4 Row and Column

3.4.1 Row

Row is the basic storage unit in the form of a string list. We fill in the methods that are given by the original framework. The main addition is a join method, which returns the join of multiple rows. It simply merges rows together with a head-to-tail approach, and supports the join method in Table.

3.4.2 Column

To reinforce the importance of the column, the *Column* class needs to record:

1. it belongs to which table,
2. its name,
3. the full name ('tableName_columnName')

With feature 3, we can solve the problem of attribute name duplication.

To support condition filtering, we need to know where the rows in the table after Cartesian product come from which table in 'from clause'. We define a 'super column' having the attribute of 'offset', which records the offset in the ultimate columns after Cartesian product. Specifically, the 'offset' attribute is derived when we do table join (Cartesian product).

3.5 Condition

We follow the instructions and fill out the framework provided by this project. In class *Condition*, we design several constructors:

1. **Condition**(Column col1, String relation, Column col2)

It is a condition representing COL1 RELATION COL2, where COL1 and COL2 are column navigators, and RELATION is one of the strings "<", ">", "<=", ">=", "=", or "!=".

2. **Condition**(Column col1, String relation, String val2)

An overloaded Constructor that calls Condition(Column col1, String relation, Column col2), used when the third parameter is a literal value.

3. **boolean test**(Row... rows)

Assuming that ROWS are rows from the respective tables from which columns are selected, returns the result of performing the test denoted. Only allows non-numeric type when checking equality or inequality. Any attempts applying "<", ">", "<=", or ">=" between non-numeric types will throw *NumberFormatException* error.

4. **static boolean test**(List<Condition> conditions, Row... rows)

This function will iteratively call boolean test(Row... rows). Return true if and only if ROWS satisfies all CONDITIONS.

4.1 Result

```

Last login: Tue Dec 20 22:44:15 on tty000
db61b ~ - /start.sh - /start.sh - java - start.sh - 159x48
db61b git:(qlmatt-dev) * ./start.sh project-diana-candy-superTEAM/qlmatt-dev
DB618 System: Version 2.0.
> load students;
Successfully loaded students.db
> load enrolled;
Successfully loaded enrolled.db
> load schedule;
Successfully loaded schedule.db
> select * from students;
SID Lastname Firstname SemEnter YearEnter Major
102 Chan Valerie S 2004 LSUnd 104
103 Xavier Jonathan S 2004 LSUnd
104 Armstrong Thomas F 2003 EECs 102
105 Brown Shana S 2004 EECs
106 Chan Yangfan F 2003 LSUnd
101 Knowles Jason F 2003 EECs
> select * from students, enrolled, schedule;
students_SID Lastname Firstname SemEnter YearEnter Major enrolled_SID enrolled_CCN Grade schedule_CCN Num Dept
103 Xavier Jonathan S 2004 LSUnd 104 21085 A- 21105 54 Math 1-2MWF 1 Pimental S 2004
104 Armstrong Thomas F 2003 EECs 102 21231 A 21228 61A EECs 2-3MWF 1 Pimental F 2003
106 Chan Yangfan F 2003 LSUnd 101 21105 B+ 21229 61B EECs 11-12MWF 155 Dwinelle F 2003
103 Xavier Jonathan S 2003 LSUnd 106 21081 B 21229 61B EECs 11-12MWF 155 Dwinelle F 2003
105 Brown Shana S 2004 EECs 104 21005 A- 21005 1A English 230-5TuTh 130 Wheeler S 2004
105 Brown Shana S 2004 LSUnd 106 21103 A 21228 61A EECs 2-3MWF 1 Pimental F 2003
103 Xavier Jonathan S 2004 LSUnd 106 21231 A 21005 1A English 9-10MWF 2301 Tolman F 2003
103 Xavier Jonathan S 2004 LSUnd 102 21105 A- 21229 61B EECs 11-12MWF 155 Dwinelle F 2003
105 Brown Shana S 2004 EECs 103 21005 B+ 21228 61A EECs 2-3MWF 1 Pimental F 2003
102 Chan Valerie S 2003 LSUnd 102 21229 A 21103 64 Math 1-2MWF 2050 VLBS F 2003
104 Armstrong Thomas F 2003 LSUnd 106 21231 A 21231 61A EECs 1-2MWF 1 Pimental S 2004
104 Armstrong Thomas F 2003 EECs 106 21231 A 21103 54 Math 1-2MWF 2050 VLBS F 2003
104 Armstrong Thomas F 2003 EECs 103 21005 B+ 21105 64 Math 1-2MWF 1 Pimental S 2004
106 Chan Valerie S 2003 LSUnd 102 21081 B+ 21001 1A English 9-10MWF 2301 Tolman F 2003
103 Xavier Jonathan S 2003 LSUnd 105 21228 A 21229 61B EECs 11-12MWF 155 Dwinelle F 2003
102 Chan Valerie S 2003 LSUnd 104 21105 A- 21229 61B EECs 11-12MWF 155 Dwinelle F 2003
106 Chan Yangfan F 2003 LSUnd 106 21103 A 21105 54 Math 1-2MWF 1 Pimental S 2004
103 Xavier Jonathan S 2003 LSUnd 101 21105 B+ 21105 64 Math 1-2MWF 1 Pimental S 2004
105 Brown Shana S 2004 EECs 106 21231 A 21001 1A English 9-10MWF 2301 Tolman F 2003
103 Xavier Jonathan S 2004 LSUnd 105 21081 B+ 21001 1A English 9-10MWF 2301 Tolman F 2003
101 Knowles Jason F 2003 EECs 102 21229 A 21232 61B EECs 1-2MWF 2050 VLBS F 2004
106 Chan Yangfan F 2003 LSUnd 103 21085 B+ 21105 64 Math 1-2MWF 1 Pimental S 2004
104 Armstrong Thomas F 2003 EECs 104 21005 A- 21229 61B EECs 11-12MWF 155 Dwinelle F 2003
103 Xavier Jonathan S 2004 LSUnd 104 21085 A- 21005 1A English 230-5TuTh 130 Wheeler S 2004
105 Brown Shana S 2004 EECs 102 21231 A 21103 54 Math 1-2MWF 2050 VLBS F 2003
102 Chan Valerie S 2003 Math 102 21105 A- 21001 1A English 9-10MWF 2301 Tolman F 2003
101 Knowles Jason F 2003 EECs 101 21228 B 21228 61A EECs 2-3MWF 1 Pimental F 2003

> select SID, Firstname
...from students
...where Lastname = 'Chan';
SID Firstname
106 Yangfan
102 Valerie

> select Firstname, Lastname, Grade
...from students, enrolled
...where students_SID = enrolled_SID and CCN = '21001';
Firstname Lastname Grade
Valerie Chan B+
Shana Brown B+
Jason Knowles B+
Yangfan Chan B

> create table enrolled2 as
...select SID from enrolled, schedule
...where enrolled_CCN = schedule_CCN and Dept = 'EECS' and Num = '61A';
> print enrolled2;
SID
21001
21002
21003
21004
21005
21006
21007
21008
21009
21010
21011
21012
21013
21014
21015
21016
21017
21018
21019
21020
21021
21022
21023
21024
21025
21026
21027
21028
21029
21030
21031
21032
21033
21034
21035
21036
21037
21038
21039
21040
21041
21042
21043
21044
21045
21046
21047
21048
21049
21050
21051
21052
21053
21054
21055
21056
21057
21058
21059
21060
21061
21062
21063
21064
21065
21066
21067
21068
21069
21070
21071
21072
21073
21074
21075
21076
21077
21078
21079
21080
21081
21082
21083
21084
21085
21086
21087
21088
21089
21090
21091
21092
21093
21094
21095
21096
21097
21098
21099
21100
21101
21102
21103
21104
21105
21106
21107
21108
21109
21110
21111
21112
21113
21114
21115
21116
21117
21118
21119
21120
21121
21122
21123
21124
21125
21126
21127
21128
21129
21130
21131
21132
21133
21134
21135
21136
21137
21138
21139
21140
21141
21142
21143
21144
21145
21146
21147
21148
21149
21150
21151
21152
21153
21154
21155
21156
21157
21158
21159
21160
21161
21162
21163
21164
21165
21166
21167
21168
21169
21170
21171
21172
21173
21174
21175
21176
21177
21178
21179
21180
21181
21182
21183
21184
21185
21186
21187
21188
21189
21190
21191
21192
21193
21194
21195
21196
21197
21198
21199
21200
21201
21202
21203
21204
21205
21206
21207
21208
21209
21210
21211
21212
21213
21214
21215
21216
21217
21218
21219
21220
21221
21222
21223
21224
21225
21226
21227
21228
21229
21230
21231
21232
21233
21234
21235
21236
21237
21238
21239
21240
21241
21242
21243
21244
21245
21246
21247
21248
21249
21250
21251
21252
21253
21254
21255
21256
21257
21258
21259
21260
21261
21262
21263
21264
21265
21266
21267
21268
21269
21270
21271
21272
21273
21274
21275
21276
21277
21278
21279
21280
21281
21282
21283
21284
21285
21286
21287
21288
21289
21290
21291
21292
21293
21294
21295
21296
21297
21298
21299
21300
21301
21302
21303
21304
21305
21306
21307
21308
21309
21310
21311
21312
21313
21314
21315
21316
21317
21318
21319
21320
21321
21322
21323
21324
21325
21326
21327
21328
21329
21330
21331
21332
21333
21334
21335
21336
21337
21338
21339

```

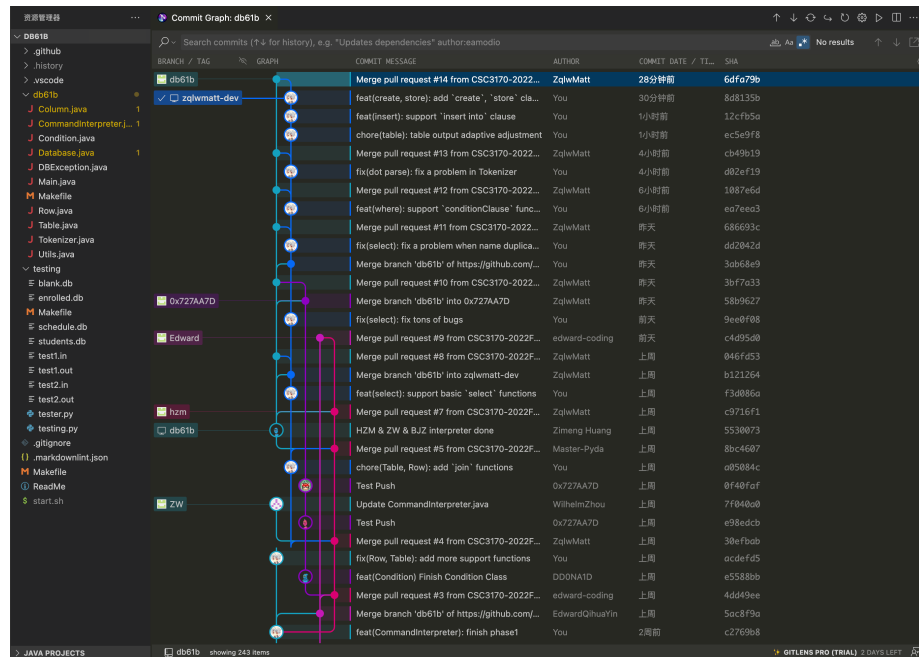
4.2 Highlight

In addition to the features mentioned in the design idea, we also made optimization in these aspects:

1. **Table.toString**: print table with adaptive adjustment.

2. **Table.select**(List<String> columnNames, List<Condition> conditions): supports column names and conditions filtering, and we increase the robustness of the function.
3. **Column**(String name, Table... tables): If the length of tables = 1, we are constructing a database table. Otherwise, we are constructing a super column (robustness).
4. **Column._offset**: use offset in super columns to derive the data after Cartesian product.

4.3 Timeline



5 Contribution

Each member of our group contribute equally on both report and presentation in this project. We follow the instructions in project description and divide tasks in six parts, everyone is responsible for one of them.

Name	Work in charge
Jingzhi Bao	whole project design and coding, presentation preparation and report writing
Zimeng Huang	<i>Command Interpreter</i> part, presentation preparation and report writing
Wei Zhou	<i>Command Interpreter</i> part, presentation preparation and report writing
Ruiyi Wang	<i>Table</i> part, presentation preparation and report writing
Jiaming Wang	<i>Table</i> part, presentation preparation and report writing
Ningyuan Wang	<i>Condition</i> part, presentation preparation and report writing
Qihua Yin	<i>Row</i> part, presentation preparation and report writing