Reed Hampton | Grant Hruzek | Hanzhi Guo

Post Production Notes

Summary

This project presented many difficulties that we had to overcome, many of these are detailed in the below section on "*Road Blocks and Difficulties*". Implementing even a partial SQL clone proved to be more difficult than we could have imagined but was also very educational at the same time. The programs production could be broken down into 5 unique steps:

1. Initial Design
2. Parser Implementation
3. Database Engine Implementation
4. Parser and Database Engine Linking
5. Testing and Bug fixing

Work Load Distribution

We decided to designate each team members workload in the implementation of the DB Engine based on function implementation.

- *This can be seen in our Function Delegation Page*.

With that in mind the team members actually implementing said functions did change from our initial delegation to guarantee the core aspects of the Database Program were functional. Before breaking down the Design/Documentation Contributions, it is pertinent to clarify that the work load was not evenly distributed among all team members.

Design/Documentation Contributions

Team Member	Contributions
Reed Hampton	Finalized Design Document, Wrote final report, Major DBParser & DBEngine Design/Implementation, Quality Assurance
Grant Hruzek	Initialized Design Document, Major DBParser & DBEngine Design/Implementation, XML/CSV Input file Design, Unit Testing, Integration Testing, Quality Assurance
Hanzhi Guo	Initial Design Document Development, Unit Testing

Design Alterations

There were many alterations to our original design, mainly in the realm of representing the data files as tables in our program. Below we have listed some of the design alterations made throughout the process:

- We initially believed we would be making our own Parser
    - **Ended up using ANTLR as an automatic parser generator**
- Initially planned to access the file with every query and command
    - Decided to implement local memory storage for tables to decrease access and update times
        - This included a "*Temporary Table Storage*" for the results of queries
        - This drastically reduced run time
- We initially planned to implement nested queries
    - This proved too difficult given the time constraints and workload
    - Discussed more in the following section

Roadblocks and Difficulties

There were **MANY** challenges during this process, here are the important challenges:

1. **Time constraints**
    - There was simply not enough time allocated to complete this project entirely
2. **Workload Distribution**
    - Evenly distributing the workload (mainly the coding) proved to be a major hindrance on pace of completion
    - Two team members implemented nearly all of the code and were unable to finish the whole project by themselves
3. **Linking Parser to Engine**
    - Finding a time/space efficient way of managing queries and stepping through the parse tree was difficult
        - Especially since we had no experience with ANTLR
4. **Implementing a Temporary Table System**
    - Some tables, the results of some queries, needed to be stored in local memory but not saved
        - This implementation took some time to get stable and repeatable
5. **Nested Queries**
    - Nested Queries proved to be extremely difficult
        - For example, calling Select on a Select statement. This requires multiple varieties of each function with different arguments as inputs
6. **Nested Conditionals**
    - Nested conditionals were similarly difficult
        - This project also required the implementation of a basic expression parser to handle conditionals
        - This **ALONG WITH** all of the other requirements was extremely difficult to complete
7. **Parsing provided Class Data**
    - Parsing this data to get into our database engine was difficult as well
        - The provided data was not uniform in its demarcation
            - This makes it extremely difficult to parse accurately
8. **Miscellaneous**
    - Some instructions were very vague
        - Particularly the third milestone
    - Scheduling meeting times and dealing with Merge conflicts are always an issue
    - Many, many more difficulties!!!

Solutions

Here we will detail our solutions to the above 7 problems. Some of these solutions may not have solved the problem but either mitigated, or worked around the issue.

1. **Time constraints**
    - There was little that could be done about this besides many sleepless nights
        - See the Development Log for a breakdown of hours spent on project
2. **Workload Distribution**
    - There was only so much that could be done about this issue.
        - The code was split as fairly as possible between the team members initially, but was soon re-divided to guarantee it was completed on time
3. **Linking Parser to Engine**
    - Decided to create a custom "Listener" class which followed up the exit path of the generated parse tree
        - This allowed us to start from the leaves and combine upwards
        - This implementation saved much head-ache and guaranteed right input
    - The custom Listener was implemented mainly using exit functions and a global stack for keywords and operators
4. **Implementing a Temporary Table System**
    - This was implemented through the creation of a Relation Table class and an Attribute sub-class
        - By making these into objects we were able to create an ArrayList which stored our temp tables
        - These could then be saved in-memory and moved into permanent storage if so desired
5. **Nested Queries**
    - Was never fully implemented but a work around was presented
        - Each nested query can be run independently and its results stored in the temp table local storage
        - The outer query could then be run on the temp table in storage for an **identical result**
6. **Nested Conditionals**
    - Believed to be handled by Expression Parser
        - We were not able to do a full testing suite due to time constraints
7. **Parsing provided Class Data**
    - This was resolved by converting the xml file input to a .csv for parsing
        - It was then parsed using the commas as delimiters and loaded into DB

Lessons Learned

There were many valuable lessons learned throughout this project. We determined they could be sub-divided into 2 categories, soft skills and hard skills.

- **Soft Skills**
  - Communication
    - We learned how vital communication is to be successful in a team-environment developing under the agile method.
    - This ranged from meeting times, to coding, to merging conflicts
  - Delegation
    - Delegation has many facets to it that you must consider, including:
      - Team members workload outside of this project
      - Upcoming deadlines, Team member technical ability, etc.
  - Expectations
    - We learned to manage our expectations of ourselves and teammates.
  - Pro-activity
    - We learned that you should **ALWAYS** be working.
- **Hard-Skills**
  - Java
    - Some of us were not experienced with Java. This project changed that
  - SQL
    - Implementing even an SQL clone gave a greater appreciation for the language and database system as a whole
  - ANTLR
    - We initially were going to go implement our own parser
    - We learned to not reinvent the wheel, and utilized ANTLR to parse
  - Good Design Practices
    - We learned good design practices ranging from:
      - Detailed commenting, Advance Planning, Communication

Overall, this project was extremely informative but very challenging. We're glad to have had the opportunity to work on it, we learned so much. Thank you very much for spending some of your valuable time to learn about our SQL-DBMS Project!

Reed Hampton | Grant Hruzek | Hanzhi Guo

DB Engine Function Delegation

Team Member	Function Name	Completed?
Reed Hampton	Select	YES (nested/multi-select issue)
Reed Hampton	Rename	YES
Reed Hampton	project	YES (multi-commands not tested)
Reed Hampton	Temporary Table System	YES (for open, write, show, select)
Reed Hampton	Math Parser/Solver	YES
Reed Hampton	Condition	YES (with one known edge case)
Reed Hampton	Conjunction	YES (with one known edge case)
Reed Hampton	Comparison	YES (tested)
Reed Hampton	Identifier	YES (tested)
Reed Hampton	Type	YES (tested)
Reed Hampton	Integer	YES (tested)
----------	----------	----------

Team Member	Function Name	Completed?
Grant Hruzek	Typed Attribute List	YES (tested)
Grant Hruzek	Attribute List	YES (tested)
Grant Hruzek	Exit	YES (tested)
Grant Hruzek	Show	YES (tested)
Grant Hruzek	Create	YES (tested)
Grant Hruzek	Write	YES (tested)
Grant Hruzek	Close	YES (tested)
Grant Hruzek	Open	YES (tested)
Grant Hruzek	Insert	YES (tested)
Grant Hruzek	XML Import	YES (tested)
Grant Hruzek	Update	NO
Grant Hruzek	Delete	NO
----------	----------	----------
Hanzhi Guo	Union	NO
Hanzhi Guo	Difference	NO
Hanzhi Guo	Product	NO

Reed Hampton | Grant Hruzek | Hanzhi Guo

Testing Suite

We have two testing suites shown here.

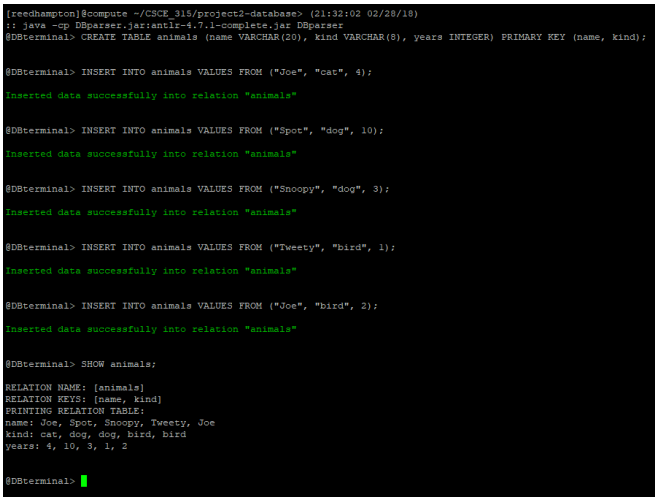
- Unit Tests
    - Small scale database tests
- Scaled tests
    - These are tests run on much larger datasets

Unit Testing

We began by testing some of the basic functionality of our Database engine on the basic SQL style commands listed below.

- Basic Commands
    - Create
    - Insert
    - Select
    - Rename
    - Project
    - Write
    - Open

1. This test creates our animals table and inserts some values:



2. This test selects from those values and outputs the result:

```
@DBterminal> select ( kind == dog ) animals;

Enter the temp tables name:
dogs

TEMPORARY RELATION NAME: [dogs]
RELATION KEYS: [ N/A]
PRINTING TEMPORARY RELATION TABLE:
name: Spot, Snoopy
kind: dog, dog
years: 10, 3


@DBterminal> select ( years > 10 ) dogs;

Enter the temp tables name:
old_dogs

TEMPORARY RELATION NAME: [old_dogs]
RELATION KEYS: [ N/A]
PRINTING TEMPORARY RELATION TABLE:
name: <empty>
kind: <empty>
years: <empty>


@DBterminal> select ( kind == cat || kind == dog ) animals;

Enter the temp tables name:
cats_or_dogs

TEMPORARY RELATION NAME: [cats_or_dogs]
RELATION KEYS: [ N/A]
PRINTING TEMPORARY RELATION TABLE:
name: Joe, Spot, Snoopy
kind: cat, dog, dog
years: 4, 10, 3


@DBterminal>
```

3. This test shows us taking a projection of some columns and then renaming them:

```
@DBterminal> project ( name , kind ) animals;

Enter the projected tables name:
tempA

TEMPORARY RELATION NAME: [tempA]
RELATION KEYS: [ N/A]
PRINTING TEMPORARY RELATION TABLE:
name: Joe, Spot, Snoopy, Tweety
kind: cat, dog, bird


@DBterminal> rename ( aname , akind ) tempA;

Succesfully changed column "name" to "aname"
Succesfully changed column "kind" to "akind"

@DBterminal> SHOW tempA ;


TEMPORARY RELATION NAME: [tempA]
RELATION KEYS: [ N/A]
PRINTING TEMPORARY RELATION TABLE:
aname: Joe, Spot, Snoopy, Tweety
akind: cat, dog, bird


@DBterminal>
```

4. Finally we end our Unit Tests with some more functionality testing over time and temporary table:

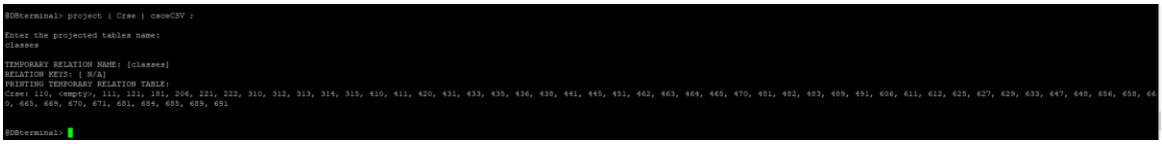
```
@DBterminal> select ( kind == cat ) animals;

Enter the temp tables name:
tempB

TEMPORARY RELATION NAME: [tempB]
RELATION KEYS: [ N/A]
PRINTING TEMPORARY RELATION TABLE:
name: Joe
kind: cat
years: 4


@DBterminal> project ( name ) tempB ;

Enter the projected tables name:
commonNames

TEMPORARY RELATION NAME: [commonNames]
RELATION KEYS: [ N/A]
PRINTING TEMPORARY RELATION TABLE:
name: Joe
```

Scaled Tests

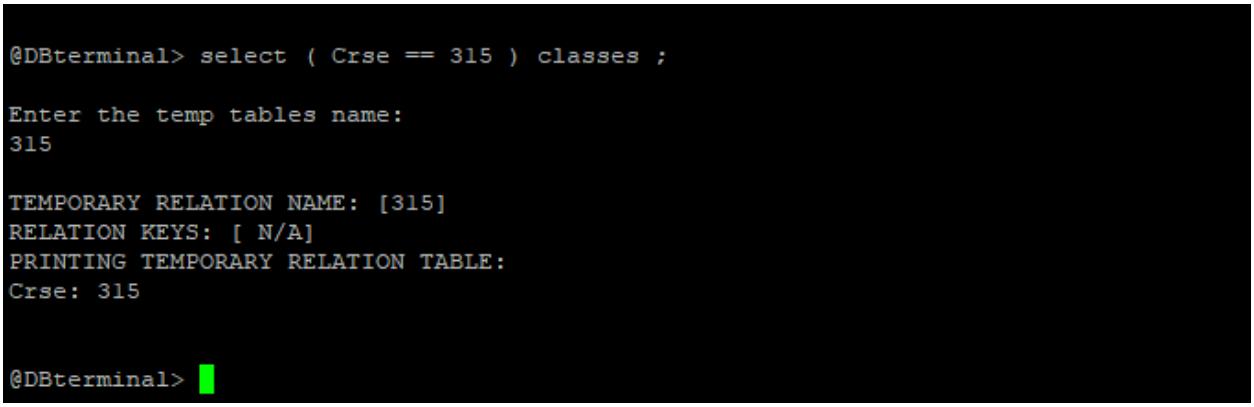
Below are some test cases from running our Database Engine on the full TAMU-CSCE Roster as proof of scalability.

1. This shows all of the rosters input (Poorly formatted terminal output):

2. This shows a projection of just the classes from this roster:



3. Finally, this image shows us selecting our class from that list:



Development Log

Date	Time	Name	Task
2/6/2018	1:20 - 2:00 PM	Hanzhi Guo	Development log and Design document
2/7/2018	9:00 - 10:30 PM	Hanzhi Guo	Worked on Design document
2/7/2018	9:00 - 10:30 PM	Grant Hruzek	Worked on Design document
2/8/2018	12:00 - 1:00 AM	Grant Hruzek	Finished First Draft on Design document
2/8/2018	12:40 - 1:40 PM	Hanzhi Guo	Changed Design document
2/8/2018	12:40 - 1:40 PM	Reed Hampton	Changed Design document
2/8/2018	12:40 - 1:40 PM	Grant Hruzek	Changed Design document
2/8/2018	7:15 - 9:00 PM	Reed Hampton	Finalized major changes to Design Document
2/11/2018	2:30 - 6:30 PM	Full Team	Began work on parser using ANTLR/user interface
2/11/2018	8:00 - 10:30 PM	Reed Hampton	Finalized grammar and parser/lexer in ANTLR
2/12/2018	8:00 - 12:00 PM	Grant Hruzek	Created DBterminal for user-input DB commands
2/13/2018	12:00 - 2:00 AM	Grant Hruzek	Connected DBterminal to ANTLR API
2/13/2018	7:00 - 9:30 PM	Full Team	Added finishing touches for Parser Submission
2/21/2018	11:00 AM - 1:40 PM	Full Team	Began DB Engine and delegated functions
2/21/2018	3:00 PM - 6:00 PM	Grant Hruzek	Implemented TypedAttList/AttList/Exit commands
2/21/2018	6:00 PM - 12:00 PM	Grant Hruzek	Implemented relationTable & attribute ADT
2/22/2018	09:00 AM - 2:40	Reed	Implemented DB Engine Functions

Date	Time	Name	Task
	PM	Hampton	
2/22/2018	3:00 PM - 10:00 PM	Grant Hruzek	Implemented CREATE/SHOW commands
2/22-23/2018	09:00 PM - 1:06 AM	Reed Hampton	Implemented 1/2 of DBEngineConcept part 3 submission
2/23/2018	03:30 PM - 6:05 PM	Reed Hampton	Finished and submitted DBEngineConcept for Part 3
2/23/2018	4:00 PM - 6:30 PM	Grant Hruzek	Implemented WRITE command
2/23/2018	6:30 PM - 8:00 PM	Grant Hruzek	Implemented OPEN/CLOSE commands
2/23/2018	10:30 PM - 11:00 PM	Grant Hruzek	Implemented INSERT (basic)
2/23/2018	8:00 PM - 11:00 PM	Hanzhi Guo	Attempted to implement UNION and DIFFERENCE
2/23/2018	10:40 PM - 11:23 PM	Reed Hampton	Updated ReadMe and submitted the Part 3 Milestone
2/26/2018	11:30 PM - 11:50 PM	Grant Hruzek	Fixed an error with the INSERT command
2/27/2018	06:10 AM - 10:45 AM	Reed Hampton	Implemented one column select support
2/27/2018	12:35 PM - 5:10 PM	Reed Hampton	Began and finished multi column select support
2/27/2018	7:00 PM - 8:00 PM	Grant Hruzek	Fixed several error cases for all DBcommands
2/27/2018	8:00 PM - 11:00 PM	Hanzhi Guo	Tested all finished functions
2/28/2018	1:00 AM - 4:00 AM	Grant Hruzek	Added functionality to parse XML files
2/28/2018	12:05 PM - 6:05 PM	Reed Hampton	Implemented rename, temp table system, multi select
2/28/2018	12:30 PM - 4:00 PM	Grant Hruzek	Parsed and Converted XML -> CSV

Date	Time	Name	Task
2/28/2018	4:00 PM - 6:00 PM	Grant Hruzek	Tested roster file with DBcommands
2/28/2018	06:05 PM - 6:45 PM	Reed Hampton	Implemented project and finished temp table system
2/28/2018	07:30 PM - 12:00 AM	Reed Hampton	Completed the final report