Western Washington University – CSCI Department
CSCI 330 Database Systems

SURLY { I/II } Report

Maxwell Schultz and Leif Johanson

Who is on your team and what's the division of labor?

Max - 50%

Leif - 50%

What programming language did you select and why?

Java, because its one the first languages we learned and it's the one we're both most comfortable with. Its utilities can be best utilized for this project. Its large set of built-in libraries was very useful in the large set of data structures it provided.

List libraries or programming language features you made use of?

- java.util.Scanner: Very useful for getting and formatting input. The tokenizing abilities of these language functions were very helpful.
- java.util.LinkedList: Very efficient for dynamic list behavior. A quick and easy data structure for holding different objects in an index based approach.
- java.util.stream.Collectors: Simple solution to guarantee unique elements in the output.

Deliverables

Checklist of deliverables	
Hardcopy of	I/II
This writeup	x
Zip file containing	I/II
This writeup	x
Test cases showing input/output	x
Source code	x
README.TXT *	x

Coverage - Did you complete all of SURLY Part I/II - what is missing?

version	Feature	Covered/Comment
I	Relation	IMPLEMENTED
I	Insert	IMPLEMENTED
I	Print	IMPLEMENTED
I	Heap Storage	IMPLEMENTED
I	Catalog	IMPLEMENTED
I	Delete	IMPLEMENTED
I	Destroy	IMPLEMENTED
II	Delete where … AND/OR	IMPLEMENTED
II	Select where … AND/OR	IMPLEMENTED
II	Project	IMPLEMENTED
II	Join	IMPLEMENTED
other	Import/Export, GUI, …	IMPLEMENTED

How did you implement

- **Relations** – Relations are implemented by using LinkedLists to separately store the schema and the list of tuples. Printing functionality is also built into the class.
- **Tuples** – Tuples are implemented as a LinkedList of distinct AttributeValue objects.
- **Attributes** – Attributes are simple objects with a few getters and basic primitive storage.
- **Insert** – Insert is initialized with some values about what is being inserted and to where. It then parses through the command and creates a relation to put into the database.
- **Catalog** – Catalog is implemented in multiple places, but mostly throughout the LexicalAnalyzer.java file. It's implemented in the methods of the commands that require it.
- **Destroy** – Destroy is very simple and uses a built-in function added to the SurlyDatabase class. The SurlyDatabase simply removes the first occurrence from the List.
- **Delete where / Select where** – Delete where and Select where are practically implemented in the same way. They go through two stages of parsing the input into all the "or" statements then splitting the remainder into tokenzied "and" statements. Then the tokenized input is executed through switch statements that check which operator is necessary.

- **Project** – Checks the database for relations that contain the same name as the input command then checks whether they are temp relations or not. Then uses loops to build the new relations by linearly taking out the necessary attributes and putting them in new tuples.
- **Join** – Goes through the schemas of the two input tables, and finds indexes where compared attributes are located, then uses indexes to build a new schema, and builds new tuples using the indexes found earlier.

Things you did differently (e.g., than the SURLY spec)

Limitations of the current release.

- No UI
- No command shell for live commands.

Extra features you added - e.g., going beyond the SURLY I/II spec

- Added processing for dot syntax within the join command.
- Building a nice and clean formatting system for outputting data on the command line.

Things you are especially proud of

- We are especially proud of the command line formatting.

Recommendations (part 2 only)

Things you would do differently if starting over now.

- We'd standardize our code base more, and try to keep it more organized while working on the project.
- We'd encapsulate our program more.

What did you learn about databases from SURLY?

- That database can be implemented in many ways as long as it outputs the same functionality to the user.
- Databases require many parts and do massive calculations to process through vast amounts of information.

Any other comments?

- N/A