**Western Washington University – CSCI Department**

**CSCI 330 Database Systems**

# SURLY II Report

Martin Smith, Eric Anderson

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

Martin Smith, Eric Anderson

We divided work a little more this time. Martin got most of the JOIN logic implemented, Eric did PROJECT. We figured out the WHERE clause and temp relation logic together. We were still using Martin’s original base file, so he typically created new Java classes. Eric formatted design, test cases, etc. Overall, at least half of the work was paired.

## What programming language did you select and why?

SURLY1 was coded in Java, and we are both the most familiar with Java. So we continued using Java.

## List libraries or programming language features you made use of?

Liked before, only some of the standard packages: java.util, java.io.

Java.util package handles all the LinkedLists within the codes (of which there are many), and java.io handles the couple of I/O exceptions. We frequently called size and add methods. Occasionally we used the Integer and Character objects for type validation.

## Deliverables

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

* \* include at top level a file titled README.TXT that provides *Installation and Demo Instructions* containing instructions on how to install and demo your program

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

|  |  |  |
| --- | --- | --- |
| **version** | **Feature** | **Covered/Comment** |
| I | Relation | Completed last month |
| I | Insert | Completed last month |
| I | Print | Completed last month |
| I | Heap Storage | In effect |
| I | Catalog | Completed last month |
| I | Destroy | Completed last month |
| II | Delete where … AND/OR | Completed, including logical operators |
| II | Select where … AND/OR | Completed, including logical operators |
| II | Project | Complete |
| II | Join | Complete, including qualify attributes |
| other | Import/Export, GUI, … | Not required for SURLY II |

## How did you implement

* **Relations** – (See SURLY 1 Report)
* **Tuples** – (See SURLY 1 Report)
* **Attributes** – (See SURLY 1 Report)
* **Insert** – (See SURLY 1 Report)
* **Catalog** – (See SURLY 1 Report)
* **Destroy** – (See SURLY 1 Report)
* **Delete where** –
* **Select where** –
* **Project** –Similar logic as Select Where. We created ProjectParser.java, which most importantly contains addInfo() and addRelation(). AddInfo() double checks if the inputted attribute/relation info matches the requested base relation, and creates a new schema and temp relation based on the specified attributes. AddRelation parses the tuples in the order specified and builds a new set of attributes/tuples, which are then inserted into the database. Duplicate tuples are erased.
* **Join -**
* **Temp Relations** – Very simple implementation – we added a boolean isTemp variable to the Relation class that is set true if the tempBuff() function is called. Various conditionals in our LexicalAnalyzer quickly check if true or false via a getter function. Often if it is true, code is skipped (i.e. won’t destroy/delete/insert into a temp relation).

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

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

Created a Parser class that does all the parsing for the Parser classes (RelationParser,

InsertParser, DeleteParser, PrintParser, DestroyParser), so that there wouldn’t be redundant code

within the Parsers.

### Things you are especially proud of:

The organization of all of the class files!

## Recommendations

### Things you would do differently if starting over now.

### What did you learn about databases from SURLY?

Databases require a lot of LinkedLists to work properly, at least in the scope of this course. A lot of conditionals and error exceptions are necessary to make the program more ergonomic, which can be computationally expensive if not done properly.

### Any other comments?

Thanks for giving us a meaningful, progressive paired assignment. It helped us see part of what is required in the workplace.