**Southeastern Louisiana University – CSIT Department**

**CMPS 439/539 Database Systems**

# SURLY II Report [[1]](#footnote-1)

Matthew Puneky & Lee White

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

Matthew Puneky – Lexical analyzer, relational algebra expressions, error handling and error reporting, all-or-nothing implementation, (de?)hydrators, some translated CRUD.

Lee White – Database design, SURLY implementation, command based CRUD implementation for relationship, attribute, and tuple.

## What programming language did you select and why?

C#, because it’s one of the programming languages that we are most familiar with.

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

Exception Handling

## Deliverables

|  |  |
| --- | --- |
| **Checklist of deliverables** |  |
| Hardcopy of | I/II/III |
| This writeup | x |
| Test cases showing input/output | x |
| Code w comments | x |
|  |  |
| Zip (or CD) containing | III |
| This writeup | x |
| Test cases showing input/output | x |
| Code w comments | 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 SURLY II

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

|  |  |  |
| --- | --- | --- |
| **version** | **Feature** | **Covered/Comment** |
| I | Relation | Covered |
| I | Insert | Covered |
| I | Input | Covered |
| I | Delete where T |  |
| I | Print | Covered (By displaying on view) |
| I | Destroy | Covered |
| I | Index - Heap |  |
| II | Index - Tree |  |
| II | Index - Hash |  |
| III | Project |  |
| III | Join | Covered |
| III | Select where … AND/OR | Covered |
| III | Delete where … AND/OR |  |
| III | Export/Import in XML |  |
| optional | relexpr , Assign, Copy, View |  |

## How did you implement

### SURLY I

* **Fields** – Implemented as a collection that contained a value and the controlling attribute for that column
* **Tuples** – Implemented as a collection that contained a collection of fields
* **Relations** – Implemented as a base unit which stored a collection of tuples, a collection of attributes, and a name for the relation.

### SURLY II

* **Heap index** –
* **Hash index** –
* **Tree index** –

### SURLY III

* **Project** –
* **Join** – Recursive method to determine how join(s) two relations should be handled, working form the inside out. Each relation that is used for the JOIN may be an expression, allowing you to next JOINS and SELECTS inside one another. The join will also take two attribute lists, identical in length, and compare them in pairs to form a temp table. The second list of attributes, or the ones to the left of the AND for the OVER statement, will have their attribute and column destroyed for temp table. The syntax is, for each relation R and attribute list T then “JOIN (R AND R) OVER (T AND T)”. EXAMPLES AT THE BOTTOM OF THIS DOCUMENT.
* **Select where** – Recursive method that resided alongside the join statement. It accepts a single relation (be it an expression or from the database) and then prunes out rows that do not match the given parameters. The select statements WHERE may take any number of AND and OR Boolean operators. For each comparison AND will take precedence over OR. This was achieved by reading for an OR, while reading for a single OR the program will read for multiple ANDS. While reading for the AND, each comparison will be evaluated and the bool output will be stored into a list of bools. Once an OR is seen, this list of bools is searched for any FALSE values and if one is found a FALSE value is saved to a secondary set of bools, if not then TRUE is stored. Once the closing “)” is found, then all of the bools in the outer most list of bools is compared linearly and if even one TRUE is found then the tuple is accepted. EXAMPLES AT THE BOTTOM OF THIS DOCUMENT.
* **Delete where** –
* **Export/Import** –

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

### Limitations of the current release.

Cannot DELETE multiple relations at a time. I’m not sure if this was even required though. Error handling can be a little off at times. The biggest limitation is that the JOIN and SELECT expressions must be encapsulated into one another, directly outputting the relation into the next statement. The output cannot be stored to a variable and used again later, but it can be used immediately so it will still work normally.

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

* Due to this being a web application, it has a user interface, which makes navigation a lot easier, and provides a better experience when using the application.
* Error handling proves information that explains why the SURLY query was rejected, the line it was rejected at, and the symbol responsible for the rejection.
* All-or-nothing was added. If you were halfway through a batch file run, it will not add everything before the failure, it will only add if there were no errors.

### Things you are especially proud of

* Error handling. This was a major feat to attempt for a first time (at this level of reporting). Try and screw up some of the test files I send you to see how the error handling works.
* The recursive method for the relational algebra was rather awesome. It may look bad on the surface but neither one of us has done a lot of recursion, so it was a great feat to get it working.
* Translating the project from “clicks” to a command system drastically increased the amount of work we had between Assignment I and II, but we still got it done! For future reference, command is actually more enjoyable and more powerful in our opinion, so probobaly dissuade people from what we did for Assignment I.

## Recommendations

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

Haha, already did that. If we were to start over again we would further separate the layers. It ended up with the BLL and DAL being combined. In the end it became a tornado of code but it is still… sorta… readable? Also, if I had a second chance, I would dropped some of the SURLY syntax for SQL style syntax.

### Did SURLY meet your objectives for this course?

Yes. It was a challenge programming a database when we have had only light interactions with databases. Definitely taught us a lot about how to handle a database.

### Suggestions on how to improve SURLY I/II assignment

More direction. What the project was supposed to do was clear, how we could go about achieving it was a little ambiguous. Sometimes more constraints are useful.

### Suggestions on how to improve the course?

More in-class activities, those were the most helpful days when I was lost.

### Any other comments?

Best. Database. Professor. Well, I’m comparing that to the nightmare stories I heard before about this class. This class was far more enjoyable than what I had thought it was going to be.

1. In Word, turn on *View/Document Map* to see an outline of this document. You can move to another section by mousing that section on the document map. To delete a footnote, select the footnote number and *SHIFT-Delete*. [↑](#footnote-ref-1)