Western Washington University – CSCI Department
CSCI 330 Database Systems

Final SURLY Report

Corben Roszak, Jess Avery

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

The team consists of Corben Roszak and Jess Avery. Jess' code was used from SURLY0 and Corben created the new objects and classes that were needed for SURLY1. Jess handled the insert parser addition, CATALOG creation and adding constants as well as the error handling related to those and general code cleanup. Corben worked on creating Delete/Destroy parsers, printing functions, and the function call changes to LexicalAnalyzer. As other problems rose they were handled by one of us.

What programming language did you select and why?

We selected Java because that is what we both have experience in and what we had programmed SURLY0 in. There was no reason to change languages.

List libraries or programming language features you made use of?

- Java.util. {LinkedList, Scanner, Arrays, List, Function.Predicate/BiPredicate}
- Java.io {File, FileNotFoundException}
- Java.Lang {IllegalArgumentException}

How did you implement

- **Relations** – Relations is an object with three fields: a name, a LinkedList of Attributes and a LinkedList of Tuples. The field are set in construction and there are no other ways to manipulate them.
- **Tuples** – Tuples is an object with a single LinkedList of Attribute Values. The object is called and passed in a LinkedList of Attribute Values with no way to manipulate the list but the value can bee requested with the right attribute name.
- **Attributes** – Attributes are handled by an object which has three fields: a name, data type and length. Each of these values can be requested, Attributes are what make up a Relation object.
- **Insert** – All of the error checking is taken care of in LexicalAnalyzer where the current database is easily accessed. If those are passed then Relation is called to insert the tuple.

- **Catalog** – Catalog is a private function in SurlyDatabase that is called when the database is first initialized. Every time a relation is created it is added to the database and when a relation is destroyed it is removed from the catalog.
- **Destroy** – Destroy is handed by the SurlyDatabase class. When a relation is destroyed it is also removed from the Catalog.
- **Delete where** – Delete where is handed by the delete parser which takes advantage of predicates to sort through the tuples and select which are then deleted. It can only be preformed on a base relation. A new temporary relation is created, which is a subclass of relation but prevents delete, insert and destroy.
- **Select / Select where** – Select and select where are handled by the same code logic taking advantage of predicates to sort tuples and add only the tuples as indicated by the user. A new temporary relation is created, which is a subclass of relation but prevents delete, insert and destroy.
- **Join** – Join is handled similarly to select and delete but has it's own special parser which is specialized to handle the join condition. A new temporary relation is created, which is a subclass of relation but prevents delete, insert and destroy. Join also checks if qualifiers are needed and handles them if they are needed.

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

Limitations of the current release.

The commands are case sensitive. The database is also inefficient in certain spots when querying tuples.

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

The main surly database was converted to a singleton to make it easier to query the database without needing to pass it to each function that required it. The conditional parser is incredible and very robust for finding the necessary tuples.

Things you are especially proud of

When reading a file the scanner reads a command even if it is disjointed or there are multiple commands on a single line. Utilizing Git lab and Jetbrains IntelliJ version control for easily keeping the project up-to-date between both of our machines.

Recommendations

Things you would do differently if starting over now.

The readfile is slightly inconvenient for later implementation. I (Corben) would change the print in relation to something a bit cleaner. I (Jessica) would move the print method in relation to print

command. I would add keys to the tuples in the form of an ID key to make indexing the tuples quicker and easier.

What did you learn about databases from SURLY?

The project helped us learn more about relational algebra and in turn learn more about relational databases. Creating the database pieces that get put together was a comprehensive lesson on databases and what they are really composed of and destroyed the preconception of the ‘black box’ of databases.

Any other comments?

Regular expressions in Java was an interesting experience.