

# Assignment 5: Putting it All Together

## Due Finals Week

In this assignment, the goal is to actually put together all of the pieces from the previous four assignments and have a working system. This will involve adding a few commands to the parser/lexer, as well as “hooking them up” so that everything works.

To get full credit on this assignment, you should have a database system that is able to be “turned on” (fired up), then process some changes and queries, and then be shut down—and have any changes that are made “stick”; that is, any new tables that are created and loaded should be remembered across runs of the program, so that updates are permanent.

For full credit, your database should implement the following commands.

### **CREATE TABLE**

This one is pretty self-explanatory. If someone gives you this command, you create the associated table and remember its schema. Here is an example of the command:

```
CREATE TABLE mytable (att1 INTEGER, att2 DOUBLE, att3  
STRING) AS HEAP;
```

The “AS” part of the command will include either “SORTED” or “HEAP”. If the type is SORTED, then there is also a required “ON” clause that tells the system what attributes to sort on:

```
CREATE TABLE MYTABLE (att1 INTEGER, att2 DOUBLE, att3  
STRING) AS SORTED ON att1, att2;
```

However, for full credit on this assignment you don’t actually have to support use of the sorted file type, though I’d encourage you to support it in your system, because all of the machinery is there!

### **INSERT INTO**

In this one, you simply bulk load (append) to the specified database table from the specified text file. The name of the text file will be given in single quotes:

```
INSERT 'myfile' INTO mytable;
```

### **DROP TABLE**

This removes the relation from the system and kills the corresponding binary file:

```
DROP TABLE mytable;
```

## **SET OUTPUT**

Basically, this tells you where the result of the output table operation at the top of the query plan goes. It is legal for a user to give the string `STDOUT` as the output location, in which case you write the output to the screen. The command is as follows:

```
SET OUTPUT STDOUT;
```

Which writes the result to the screen. Or:

```
SET OUTPUT 'myfile';
```

Finally, there is a third option:

```
SET OUTPUT NONE;
```

This option makes it so that you do not actually execute any query that you are given; you simply write out the query plan to the screen (that is, you simply give your A4.2 answer directly to the user without running it).

## **SELECT...**

Finally, the very last thing that I'll ask you to do is to actually run an SQL query that the user gives you!

## **A note regarding statistics and the TPC-H**

Naturally, your database will use the statistics object to keep track of the number of distinct values for each and every database attribute. When your database is shut down, this object is serialized and written out. When your database is fired back up, this object is re-read. However, for A5 you will not be asked to actually have the ability to update the distinct value counts that the statistics object stores (though you can optionally do this: see below). The fact that you won't actually keep track of the statistics is a bit of a problem for your A4.2/A4 demo and implementation, since you'll need this info to compile queries.

So, let's do the following. When you show up for your A4.2/A5 demo, you should have already created and loaded up the TPC-H schema. You should manually change the text file that your statistic object serializes itself to so that all of the distinct value counts that Lixia gave you for A4.1 are already in there. That way, your statistics module will have some meaningful numbers to feed into the optimizer.

Finally, if you have some extra time and are sufficiently ambitious, you might want to add the following command to your database:

```
UPDATE STATISTICS for mytable;
```

What this command does is to essentially a query for every attribute of `mytable` that first does a duplicate removal on the attribute, and then counts the number of tuples in the result set. All of the distinct value counts are then written into the statistics object.