CSCI 3357: Database System Implementation
Homework Assignment 9
Due Thursday, November 16

Note: This assignment requires a correct solution to HW 6. If you have any doubts, download and use my solution.

1. In HW 6 you modified the record manager to handle null values. Your first task is to modify the class `Constant` to implement a "null constant". In particular:

- Add a constructor to the class to create the null constant. Let's assume that a null constant has null values for both `ival` and `sval`. Then this constructor is trivial — it has no argument and have it do nothing (since `ival` and `sval` are null by default).
- Add the method `isNull()` to class `Constant` that returns true if the object is a null constant and false otherwise.
- Modify the methods `equals`, `compareTo`, `hashCode`, and `toString`. A null constant will never compare successfully with another object (even another null constant!). That is, if you have a null constant, calling the `equals` method should always return false, calling the `compareTo` method should return -1, calling `hashCode` should return 0, and calling `toString` should return "null".

2. Modify the `TableScan` class so that the method `getVal` will return a null constant if the value of the requested field is null, and the method `setVal` will set the requested field value to null if its argument is a null constant.

3. The next task is to modify the class `Term`. Currently, a term must be of the form "`e1=e2`" for expressions `e1` and `e2`. You need to generalize terms so that they can also be of the form "`e1<e2`", "`e1>e2`" and "`e1 IS null`". In this last term, the operator is the keyword "IS".

This class has a lot of methods, many of which are uninteresting to us. I have made appropriate modifications to those methods in the file *Term.java*, and you should download this file before you begin.

My version defines a variable `op` that holds the term's operation. Its value is one of the following four defined constants:

```
public static final int EQ=0, LT=1, GT=2, IS=3;
```

My class also defines a new constructor that has three arguments — the left-hand expression, an int denoting the operator, and the right-hand expression. The existing two-argument constructor has been modified to use the operator `EQ`.

Your job is to modify the methods `isSatisfied` and `toString`.
The `isSatisfied` method is where a term gets evaluated. Modify it so it does the appropriate comparison as specified by the operator. The `IS` operator should return true if its left-side expression evaluates to a null constant. Make sure that the other operators work correctly in the presence of null constants. Note that comparing a null constant to any other constant is always false, even if the other constant is also null.

The `toString` method constructs a representation of the term in SQL syntax, such as "A > 3", "B is null", etc.

4. Your final task is to modify the lexer and parser to handle the added functionality. Modify the lexer to have the additional keywords “null” and “is”. Modify the parser based on the following modifications to the SimpleDB grammar:

```
<Constant> := StrTok | IntTok | <NullConstant>
<NullConstant> := NULL  // i.e., the keyword "null"
<Term> := <Expression> <Op> <Expression>
<Op>  := < | > | = | IS
```

Although the grammar doesn’t mention it, the parser should enforce the constraint that if the operator is “IS”, then its right-hand-side expression must be comprised of a <NullConstant> constant.

You should download my test program *HW9Test.java*, to help you test your final code. In the early stages of debugging, I recommend that you simplify it.

One of the great things about changing the parser is that you have actually changed the language and can see the changes via JDBC. In particular, look at my JDBC client program named *HW9Client.java.* It runs in embedded mode. **Try it**. It performs the following actions, none of which were possible prior to this assignment.

- It uses an update command to set the `GradYear` value for the `STUDENT` record "amy" to be null;
- It inserts a new `STUDENT` record for "tom", whose `MajorId` value is 20 and whose `GradYear` value is null;
- It issues a query to print the names of all students graduating after 2019 and before 2022; and
- It issues a query to print the names of all students having a null grad year.

When you are done, create a zip file containing the files *Constant.java, Term.java, TableScan.java, Lexer.java,* and *Parser.java*. Then submit your zip file to Canvas.