CSCI 3357: Database System Implementation
Homework Assignment 8
Due Monday, November 18

1. In HW 6 you modified the record manager to handle null values. Your first task is to modify the query manager to also understand nulls. In particular:

- Modify the `Constant` class so that `Constant` objects that have null values for both `ival` and `sval` are treated as null constants. The class will need a third constructor to create a null constant; the easiest way to write this constructor is to give it no argument and have it do nothing. The class will also need a method `isNull()`. A null constant should never be compared with another object (even itself!), so the `equals` and `compareTo` methods need not change. The other methods should change appropriately.
- Modify the `TableScan` class so that the method `getVal` will return a null constant if the requested field value is null, and the method `setVal` will set the requested field value to null if the argument is a null constant.

2. Your 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`".

The first thing to do is to handle the constructors. In particular, you should create a new constructor that has three arguments—namely, two expressions and an int denoting the operator. For consistency, everyone should define the following constants for the four operators you need to support:

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

Note that the `ISNULL` operator does not have a right-side expression. The operator will simply ignore the value of the right-side expression passed into the `Term` constructor. For legacy use, you should also keep the existing two-arg constructor, but modify it so that it always uses the `EQ` operator.

The `isSatisfied` method is where the term gets evaluated. Modify it so it does the appropriate comparison, based on the operator. The `ISNULL` operator should return true if its left-side expression is a null constant. Modify the other operators so that they 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 `reductionFactor` method is used by the planner. A reduction factor of N means that only $1/N^{th}$ of the records will be satisfied. You can assume that the comparison operators `LT` and `GT` reduce the size of the output by 50% (a reduction factor of 2), and `ISNULL` reduces the size of the output by 90% (a reduction factor of 10).

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

The `appliesTo` method is straightforward. You just need to modify it to deal with the fact that `ISNULL` does not have a right-side expression.

The methods `equatesWithField` and `equatesWithConstant` apply only for the `EQ` operator. Modify them so that they return null with any other operator.

3. Your third task is to modify the parser to handle the added functionality. The SimpleDB grammar should be modified as follows:

```
<Constant> := StrTok | IntTok | NULL
<Term> := <Expression> <Op> [ <Expression ]
<Op>  := < | > | = | IS NULL
```

- Modify the class `Lexer` so that "null" and "is" are keywords.
- Modify the class `Parser` to implement the revised grammar.

4. 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, write a JDBC client program named *HW8Client* that:

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