Comp 430 A4 Part One

A4 Part One is worth 25% of your overall A4 grade. It is due on Friday, March 29th, at the beginning of class.

Overview

In A4, you will be implementing (in Java) a single-user database system with a fairly sophisticated query optimizer. For the first part of A4, your task is to take the Java parser and database catalog code that I have supplied you with, and extend the parser so that it performs appropriate semantic checks on the input SQL. Specifically, you need to:

select
where

1) Make sure that there are no type mismatches in any expressions. For example, it is valid to compare integers and floating point numbers, but not integers and text strings. For another example, the only arithmetic operation that is valid on a text string is a "+" (which is a concatenation)… anything else should result in an error.

from

2) Make sure that all of the referenced tables exist in the database.

select
where
group

3) Make sure that all of the referenced attributes exist, and are correctly attached to the tables that are indicated in the query.

4) Make sure that in the case of an aggregation query, the only selected attributes (other than the aggregates) must be functions of the grouping attributes.

In the case that you find any errors, you should print out a descriptive and meaningful arrow message to the screen.

Getting Started

Since this is not a compilers class, I am going to supply you with a basic SQL parser. So the first thing that you need to do is to download the .zip file I have supplied you with. You will see that it contains several Java source files, a file called `Catalog.xml`, a Java `.jar` file, and a file called `SQL.g`.

The `SQL.g` file contains a context free grammar for an SQL Java-based parser, developed using a compiler-compiler called `Antlr`. You should look over this file carefully. You can get through this project in its entirety without understanding precisely how this `SQL.g` file works and what `Antlr` is, but it is worthwhile to do some web surfing to try and figure out what is going on.

It is worth mentioning that the grammar for our restricted version of SQL has a few peculiarities. It does not allow subqueries. It allows only a single attribute in the grouping clause. The `WHERE` clause must be in CNF (conjunctive normal form), with

parens enclosing all of the disjunctions in the query. All tables listed must have an accompanying `AS`, and all attributes must be referred to using the "dot" notation. For example, here is a valid query:

```
SELECT
        n.n_name,
        SUM(l.l_extendedprice * (1 - l.l_discount))
FROM
        customer AS c,
        orders AS o,
        lineitem AS l,
        supplier AS s,
        nation AS n,
        region AS r
WHERE
        (c.c_custkey = o.o_custkey)
        AND (l.l_orderkey = o.o_orderkey)
        AND (l.l_suppkey = s.s_suppkey)
        AND (c.c_nationkey = s.s_nationkey)
        AND (s.s_nationkey = n.n_nationkey)
        AND (n.n_regionkey = r.r_regionkey)
        AND (r.r_name = "region")
        AND (o.o_orderdate > "date1" OR o.o_orderdate = "date1")
        AND (NOT o.o_orderdate < "date2")
GROUP BY
        n.n_name;
```

(As an aside, the schema used above, which is the one we will be using throughout A4, is known as the "TPC-H schema" and it is a standard, database industry benchmark; see `http://www.tpc.org/tpch/spec/tpch2.7.0.pdf` for a description of the TPC-H benchmark and check out page 12 for a picture of the schema).

To actually get the parser going, you first need to compile it into Java. To do this, make sure that you are in the directory with the `SQL.g` file and the `.jar` file I gave you (`antlr-3.2.jar`), and at the command line type:

```
java -cp antlr-3.2.jar org.antlr.Tool SQL.g
```

This will invoke `Antlr` (which is a java program, embedded in the `.jar` file), and cause it to produce two Java files: `SQLLexer.java` and `SQLParser.java`. This is a lexer and a parser that where generated from `SQL.g`. At this point, you should load all of the Java files (including `SQLLexer.java` and `SQLParser.java`) into a project using your favorite development environment (Eclipse, DrJava, etc.), and compile them (but make sure that when you do, you include `antlr-3.2.jar` in your class path!).

Once you have compiled all of the Java code, go ahead and run everything using the sample `main` I have supplied you with in `Interpreter.java`. The first thing that will happen when you run the program is that my code will print out a bunch of stuff to the screen; this is the contents of the database catalog, which the program read in by processing the `Catalog.xml` file (which corresponds to the TPC-H schema). Carefully look over the code to understand the data structures that it printed out; they contain all of the information regarding the various tables in the database, the attributes, and the types, so obviously when you do your semantic checks you'll be writing code that will look at this data structures.

Now you can type in an SQL query, that the parser will parse for you. Go ahead and copy and paste the above SQL query into the program, and then press return. The result of the parsing is a bunch of data structures that describe the query; in my sample code, they are just printed to the screen.

Finally, note that if you type `quit;` at the SQL command prompt, the program will exit.

What You Need To Do

For the first part of A4, you need to take all of this stuff that I have supplied you with and add the semantic checks described above. Basically, this means writing code that looks at the data structures containing the catalog and the data structures containing the parsed query, and makes sure that everything is semantically OK and that they match up as described above. You can use or modify any of my code however you see fit, or you can choose not to use it at all and do everything on your own.

Within a few days, I'll give you a suite of around queries (over the TPC-H schema) that you will need to run semantic checks on. To turn in A4 Part One, you'll need to print out your results and hand in a hard copy, as well as submit an electronic copy of your code.