CSCI 3357: Database System Implementation
Homework Assignment 10
Due Monday, December 9

In HW 7 you modified the query processor to implement the *union* and *rename* relational algebra operators. In this assignment you will extend these modifications into the parser and planner. Warning: This is a time-consuming assignment; start early.

1. Write classes `UnionPlan` and `RenamePlan`. Make reasonable assumptions about how to implement their statistical methods.

2. Standard SQL uses the `UNION` keyword to connect select statements (much in the same way that the `AND` keyword separates terms in a predicate). For example, the following query returns the names of students having sid 1, 3, or 5:

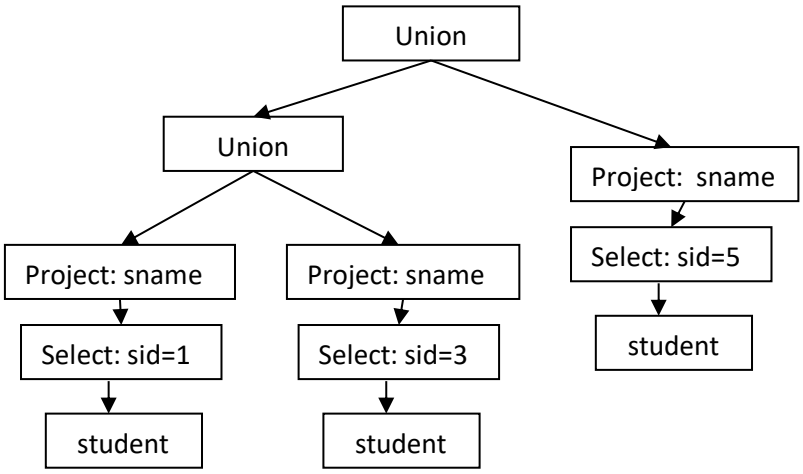
```
select sname from student where sid = 1
union
select sname from student where sid = 3
union
select sname from student where sid = 5
```

You can incorporate this new functionality into the SimpleDB grammar by adding the following rule:

```
<UnionQuery> := <Query> [ UNION <UnionQuery> ]
```

a) Modify the classes `Lexer` and `Parser` to implement this new rule. The parser method for the new syntactic category `<UnionQuery>` should return a list of `QueryData` objects—one for each subquery.

b) Modify `BasicQueryPlanner` to have a method `createUnionPlan`, which takes a list of `QueryData` objects as its argument and creates a query tree having a `UnionPlan` node for each `UNION` keyword in the query. If there are no `UNION` keywords in the query, then its plan will be the same as the plan created by the current basic query planner. For example, the plan for the above query should correspond to the following query tree:



c) Modify the `QueryPlanner` interface to have the method `createUnionPlan`, and modify the `createQueryPlan` method of `Planner` so that it calls the query planner's `createUnionPlan` method instead of `createQueryPlan`.

3. Standard SQL uses the `AS` keyword in its select clause to rename a field. For example, the following query returns the id, name and major of the students graduating in 2020, with the field `SId` renamed as `StudentNum` and the field `MajorId` renamed as `MajorDept`:

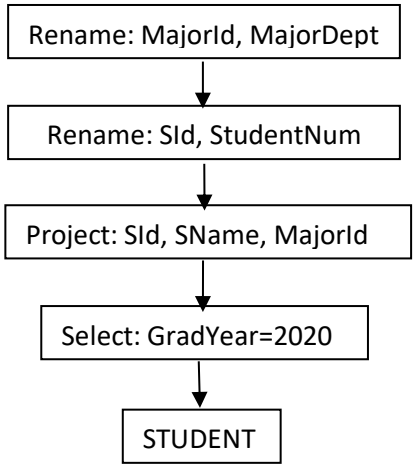
```
select SId as StudentNum, SName, MajorId as MajorDept
from Student
where GradYear = 2020
```

You can incorporate this functionality into the SimpleDB grammar by modifying the rule for `<SelectList>` and adding a rule for the new category `<SelectField>`, as follows:

```
<SelectList>  := <SelectField> [ , <SelectList> ]
<SelectField> := <Field> [ AS <Field> ]
```

a) Modify the `Parser` class to implement these grammar changes. Create a class `FieldData` to hold the values extracted from the `<SelectField>` rule. Then modify `QueryData` so that its variable `fields` is a list of `FieldData` objects instead of a list of strings.

b) Modify step 4 of the `BasicQueryPlanner` method `createPlan` to add `RenamePlan` nodes to the query plan it creates. There should be one node for each renamed field, and these nodes should appear (in any order) above the `ProjectPlan` node. For example, the plan for the above query should correspond to the following query tree:



Note that the renaming happens after the project operation occurs, which means that the where-clause can only refer to original field names, before the renaming occurs. For example, consider the query below. Note that the predicate contains the term `SId > 5` and not `StudentNum > 5`.

```
select SId as StudentNum, SName, MajorId as MajorDept
from Student
where GradYear = 2020 and SId > 5
```

Once you get everything to work, have some fun. Run the SimpleIJ client program, and execute some SQL commands that involve union and renaming. For example, try this:

```
select sname as person from student union select prof as person
from section
```