

Your task is to design, describe, and implement a method that translates *Safe Range Calculus* ( $CALC_{SR}$ ) queries into equivalent SQL queries. You may wish to use the method of in the textbook as a guide. For example, you may decide to follow that method to obtain a relational algebra expression which you then translate into SQL. However, you may wish to devise a more direct mapping that takes advantage of SQL features. For example, it may be easier to map quantification in the calculus query directly to quantification in SQL.

You can earn the full score on this assignment by implementing a method that translates a  $CALC_{SR}$  query to an equivalent SQL query, irrespective of complexity or elegance. However, you are encouraged to try to devise a method that is simple (to describe and implement) and that produces SQL queries that are short, easy to understand, and likely to execute efficiently. You may wish to include formal or informal claims to this effect in your write-up.

Your implementation must check that input queries are safe-range and flag an error if they are not. If your method or implementation is unable to handle all cases, make sure you clearly describe the limitations.

**Input Format** We will use the syntax described in the textbook for writing  $CALC_{SR}$  queries. However, we use the following plain-text representations for the components: Relation names are alphanumeric strings beginning with a letter (e.g., `Edges`, `Nodes`, `Employees234`, `Dept4b`). Variables have alphanumeric names that begin with an underscore (e.g., `_x`, `_part`). Anonymous variables are denoted using a single underscore character. The operators *and*, *or*, and *not* are written as `.and`, `.or`, and `.not`, respectively. (Note the period before the words.) Universal and existential quantification is written as `.exists` and `.forall`, respectively. The rest of the syntax is as described in the textbook, as illustrated by the following rendition of the query  $q_1$  of Example 5.4.9 from the textbook:

```
<a, x, y> : A1 A2 A3 | .exists z (P(x,y,z) .or
                        (R(x,y) .and ((S(z) .and .not T(x,z))
                        .or T(y,z))))
```

If you encounter parsing difficulties, you may modify this syntax slightly as long as you note the changes clearly and your syntax permits all  $CALC_{SR}$  queries to be expressed.

**Output Format** Your program should produce a SQL query in text form as the output. For concreteness, we will require that the query you produce executes correctly when typed at the prompt of either `sqlplus` or `psql` (indicate which, if it does not work with both). Please pretty-print your query using the conventions found in most textbooks. (At the very least, put each SQL clause on a separate line, with proper indentation for subqueries.)

**Using Others' Work** You are free to use any resources, including code, authored by someone else. However, you must **clearly and prominently** cite and acknowledge all such

resources. Further, you are responsible for knowing how everything in your submission works, irrespective of its origin. Thus, you can use other people's code, but you must understand its internals and not use it as a black box.

**Method Description** There is no hard page limit for the description of your method. Use as much space as you believe is necessary to clearly describe your work. However, please be concise and make sure you proofread your work. You will lose points for poorly written work. You must submit your work in plain text, Latex, or PDF format. (Unless you have a strong preference, please use Latex to prepare your report, and submit the .tex files along with any nonstandard supporting files you use.) Please use the `less`, `latex`, or `gv` commands on your `linuxlab` account to verify that your plain text, Latex, or PDF files, respectively, are properly displayed.

**Code** You may implement your method in C or Java. If you'd like to use some other language, check with me first. You should submit the source code and only the source code. Do **not** submit object code or byte code (.o files, .class files, or machine code). Include a make-file that processes your source files in response to a `make` command and a README file that provides brief instructions on compilation and use. Please also include a TESTS file that contains a dozen or so test queries, including the query of Example 5.4.9.

**Submission** Please submit your work electronically as a gzipped tarred file named using the scheme `LastnameIJ-hw1-NNNN.tgz` (replacing `LastnameIJ` with your last name and initials, and `NNNN` with a 4-digit number) by anonymous FTP upload to `ftp.cs.umd.edu`, directory `/incoming/chaw/724`. Please verify that uncompressing and untarring your submission produces all the source code for your implementation (including a Makefile and a README file) and a write-up of your method (in plain text, Latex, or PDF format).