National Institute of Technology Calicut

Department of Computer Science and Engineering

CS3095D DATABASE MANAGEMENT SYSTEMS LABORATORY

S5/S7 B.Tech. - Monsoon Semester 2022

Exercise No: 07

Relational Algebra

Running the RA Tool and Syntaxes

The command to run the tool using terminal is "`java -jar ra.jar`".

Then you will see a RA ("`ra>` ") prompt on the terminal, type the operation and hit enter.

You can exit RA by issuing the `\quit;` command. Use the `\list;` command to see what relations are available for query in your database.

RA tool syntaxes for Relational Algebra operators

[**Note** that every command/operator should start with a backslash (\), and every query/command should be terminated by a semicolon (;).]

- `\select_{`*cond*`}` is the relational selection operator. For example, to select `Drinker` tuples with name Amy or Ben, we can write `\select_{name = 'Amy' or name = 'Ben'} Drinker;`. Syntax for *cond* follows SQL. Note that string literals should be enclosed in ***single*** quotes, and you may use boolean operators `and`, `or`, and `not`. Comparison operators <=, <, =, >, >=, and <> work on both string and numeric types. For string match you can use the SQL `LIKE` operator; e.g., `\select_{name like 'A%'} drinker;` finds all drinkers whose name start with A, as `%` is a wildcard character that matches any number of characters.
- `\project_{`*attr_list*`}` is the relational projection operator, where *attr_list* is a comma-separated list of attribute names. For example, to find out what beers are served by Talk of the Town (but without the price information), we can write `\project_{bar, beer} (\select_{bar = 'Talk of the Town'} Serves);`.
- `\join_{`*cond*`}` is the relational theta-join operator. For example, to join `Drinker(name, address)` and `Frequents(drinker, bar, times_a_week)` relations together using drinker name, we can write `Drinker \join_{name = drinker} Frequents;`. Syntax for *cond* again follows SQL; see notes on `\select` for more details.
- `\join` is the relational natural join operator. For example, to join `Drinker(name, address)` and `Frequents(drinker, bar, times_a_week)` relations together using drinker name, we can write `Drinker \join \rename_{name, bar,`

`times_a_week} Frequents;`. Natural join will automatically equate all pairs of identically named attributes from its inputs (in this case, `name`), and output only one attribute per pair. Here we use `\rename` to create two `name` attributes for the natural join; see notes on `\rename` below for more details.

- `\cross` is the relational cross product operator. For example, to compute the cross product of `Drinker` and `Frequents`, we can write `Drinker \cross Frequents;`.
- `\union`, `\diff`, and `\intersect` are the relational union, difference, and intersect operators. For a trivial example, to compute the union, difference, and intersection between `Drinker` and itself, we can write `Drinker \union Drinker;`, `Drinker \diff Drinker;`, and `Drinker \intersect Drinker;`, which would return `Drinker` itself, an empty relation, and `Drinker` itself, respectively.
- `\rename_{`*`new_attr_name_list`*`}` is the relational rename operator, where *`new_attr_name_list`* is a comma-separated list of new names, one for each attribute of the input relation. For example, to rename the attributes of relation `Drinker` and compute the cross product of `Drinker` and itself, we can write `\rename_{name1, address1} Drinker \cross \rename_{name2, address2} Drinker;`.

End of the document