April 10, 2018

Relatex Manual

Relational Algebra LaTeX Query Executor

Relatex is a relational algebra query executor that processes queries written in LaTeX. Relations can be loaded from CSV files that specify the attribute names and types. Simple queries as well as constraint checks can be tested with relatex.

USAGE

```
relatex [-dhntv] file
```

There are several options that relatex recognizes, and are described in the next section. The file that relatex will process is specified as an argument on the command line, and needs to follow the format specified LaTeX Format section. Relatex will open the file specified and parse it regardless of the options. By default relatex will then load the CSV files into the environment that are in the specified data directory. Once the environment is loaded relatex will then execute the queries and output the result of the final assignment, or the result of the optional constraint.

OPTIONS

```
  -d (--datapath)    Specify the data directory
  -h (--help)        Display this help message
  -n (--noexecute)   Only validate the query, do not execute
  -t (--outputtree)  Output the parsed query as a tree
  -v (--verbose)     Verbose output
```

The datapath option specifies the directory where the CSV files for the environment are located. The name of the CSV file without the .csv extension will be the name of the relation that is loaded into the environment.

The help option just displays the expected usage and options.

The noexecute option only validates the query, it will not load the environment, nor will it try to execute the query. If there is an issue with the query, then it will output the appropriate error message.

The outputtree option will output the parse tree in a Lisp like fashion. This can be helpful for understanding what order of operations will take place.

The verbose option will output progress as relatex processes the query. This option will output the tree, as well as output the schemas of the relations that are loaded from into the environment.

LaTeX Format

The format of the queries are compatible with LaTeX, and the assignments and constraints are in math mode. Below is a table of equivalent relatex commands:

Description	Symbol	ReLaTeX
Relation Declaration	$Relation(attr1,attr2)$	`Relation(attr1,attr2)`
Assignment	:=	`:=`
Union	$\cup$	`\cup`
Intersection	$\cap$	`\cap`
Difference	$-$	`-`
Cartesian Product	$\times$	`\times`
Natural Join	$\bowtie$	`\bowtie`
Theta Join	$\bowtie_{Condition}$	`\bowtie_{ Condition }`
Selection	$\sigma_{Condition}$	`\sigma_{ Condition }`
Projection	$\pi_{ColumnList}$	`\pi_{ ColumnList }`
Rename	$\rho_{Relation(attr1,attr2)}$	`\rho_{ Relation(attr1,attr2) }`
OR	$\vee$	`\vee`
AND	$\wedge$	`\wedge`
Not equal	$\ne$	`\ne`
Equal	$=$	`=`
Less Than	$<$	`<`
Greater Than	$>$	`>`
Less Than Equal	$\leq$	`\leq`
Greater Than Equal	$\geq$	`\geq`
Plus	$+$	`+`
Minus	$-$	`-`
Multiply	$\cdot$	`\cdot`
Divide	$\div$	`\div`
Modulus	mod	`\bmod`
Negate	$\neg$	`\neg`
Empty Set	$\varnothing$	`\varnothing`
Comment		`% Comment`
String	*“Literal String”*	``` ``Literal String'' ```
String Escape	*&*	`\&`
	%	`\%`
	$	`\$`
	#	`\#`
	_	`\_`
	{	`\{`
	}	`\}`
	\	`\backslash`
	~	`\sim`
	^	`\hat{}`

The following query:

$S(a,b,c) \coloneqq \pi_{a,b,c}(\sigma_{a>5}(R))$

Would have LaTeX format of:

```
$S(a,b,c) := \pi_{a,b,c}(\sigma_{a>5}(R))$
```

As a note in order to separate each line in the linear notation, either a blank line must be put in between each line, or the line must end with `\\`.

CSV File Format

The relations can be stored in a CSV files. The name of the file without .csv extension will be the name of the relation loaded into the environment. Each column is a different attribute, with the header row containing the attribute name and the data type. A colon separates the data type and attribute name. Four data types are allowed in the CSV files: string, float, integer, and boolean. Rows two and on, each have one tuple in them.

The following is an example of relation R that would be in the file R.csv:

a : integer	b : string	c : float	d : boolean
3	Hello	3.4	true
4	World	1.1	false
6	Goodbye	8.8	false
7	None	9.3	true

Relational Algebra EBNF

```
Query := { Assignments } [ Constraint ]
Assignment := $ RelationDeclaration := SubQuery $
Constraint := $ SubQuery ( = RelationConstant | \subseteq SubQuery ) $
RelationDeclaration := Relation ( ColumnList )
SubQuery := IntersectSet { UDOperator IntersectSet }
Relation := Identifier
ColumnList := ColumnIdentifier { , ColumnIdentifier }
IntersectSet := CPJSet { ISOperator CPJSet }
UDOperator := \cup | -
CPJSet := SPRSet { CPJOperator SPRSet }
ISOperator := \cap
SPRSet := SPROperation | Relation | ( SubQuery )
CPJOperator := \times | \bowtie | \bowtie_{ Condition }
SPROperation := SPROperator ( SubQuery )
SPROperator := SelectOperator | ProjectOperator | RenameOperator
SelectOperator := \sigma_{ Condition }
ProjectOperator := \pi_{ ColumnList }
RenameOperator := \rho_{ Relation ( ColumnList ) }
Condition := Conjunction { OROperator Conjunction }
Conjunction := EqualityRelation { ANDOpeartor EqualityRelation }
OROperator := \vee
EqualityRelation := InequalityRelation { EQOperator InequalityRelation }
ANDOperator := \wedge
InequalityRelation := Term { IEQOperator Term }
EQOperator := \ne | =
Term := SimpleTerm { AddOperator SimpleTerm }
IEQOperator := < | > | \leq | \geq
SimpleTerm := UnaryExpression { MultOperator UnaryExpression }
AddOperator := + | -
UnaryExpression := ( UnaryOperator UnaryExpression ) | PrimaryExpression
MultOperator := \cdot | \div | \bmod
UnaryOperator := \neg | - | +
PrimaryExpression:= ( Condition ) | Constant | ColumnIdentifier
Constant := IntConstant | FloatConstant | StringConstant
```

Token Rules

```
ColumnIdentifier := Identifier { . Identifier }
Identifier := Alpha { ( Digit | Alpha ) }
IntConstant := Digit { Digit }
FloatConstant := Digit { Digit } [ . Digit { Digit } ]
StringConstant := `` { ( Digit | Alpha | WhiteSpace | Punctuation ) } ''
RelationConstant := \varnothing
Comment := % { ( Printable | Tab ) } NewLine
Digit := 0 - 9
Alpha := A - Z | a - z
WhiteSpace := Tab | CarriageReturn | NewLine | Space
Printable := Space - ~
Punctuation := ! - " | ' - / | : - @ | [ | ] | ` | | | \& | \% | \$ | \# | \_
| \{ | \} | \backslash | \sim | \hat{}
```