PCRS Relational Algebra Documentation

Relation

A relation is referenced with its name. A relation name is a well formed expression on its own. For example,

```
Question;
```

would evaluate to all of the tuples in the `Question` relation.

Relation names must begin with an alphabet character.

Attributes

An attribute is referenced by its name, or optionally using a prefixed dot notation `relation.attribute`. For example,

- `id`
- `Question.id`

both reference the `id` attribute of the `Question` relation.

The prefix allows disambiguation when multiple attributes have the same name. For example, the Cartesian product of `Question` and `Answer` would require

- `Answer.id`
- `Question.id`

and not the ambiguous reference `id`.

A relation that results from a set operation that combines tuples from two or more relations cannot be referenced using the prefix notation, without using a rename operator.

Arguments

Some relational algebra operators take additional arguments, for example Select and Project. In the syntax, arguments use a LaTeX-ish format,

```
Operator_{argument};
```

The specific operator descriptions below have more examples.

Logical Operators

Keywords: `not`, `and`, `or`

Conditional Operators

Keywords: `<`, `>`, `<=`, `>=`, `=`, `!=`, `<>`

Unary operators

Select

Keyword: `\select`

```
\select_{answer='42'} Question;
\select_{question='life' and answer='42'} Question;
```

Select evaluates each tuple in a relation against a boolean expression passed in as a required argument. The boolean expression can be composed of attribute references, numbers, quoted strings, comparison operators or logical operators. Attribute references must be valid.

Project

Keyword: `\project`

```
\project_{id} Question;
\project_{id, question} Question;
```

Project requires a comma separated list of attributes as a required argument. Attribute references must be valid.

Assignment

Keyword: `:=`

```
Q := \project_{answer} Question;
```

```
Q(a) := \project_{answer} Question;
```

Assignment assigns a new name and optionally attributes names to a relation. The relation name must not conflict with a name that already exists. The attribute names must be unambiguous. The new name can be referenced by any statement following the assignment.

Assignment uses the syntax

```
name(attribute-list) := relation;
```

where `(attribute-list)` is an optional, comma separated list of new names for all of the attributes in the relation.

Rename

Keyword: `\rename`

```
\rename_{Q} Question;
\rename_{(i, q)} Question;
\rename_{Q(i, q)} Question;
```

Rename assigns a new name to a relation, renames all of its attributes or does both. The new name must not conflict with a name that already exists. The attributes must be unambiguous.

Rename requires one argument with the syntax

```
relation(attribute-list);
```

where `relation` and `attribute-list` are both optional, but at least one is required, and `attribute-list` is a comma separated list of new names for all of the attributes in the relation.

Binary operators

Join operators

A relation cannot be joined with itself - even if select or project operator has been applied to it. To join a relation with itself, at least one of the instances of the relation must be renamed.

Cartesian Product

Keyword: `\product`

```
Question \product Answer;
\rename_{Q1} Question \product \rename_{Q2} Question;
```

Natural Join

Keyword: `\natural_join`

```
Question \natural_join Answer;
```

Theta Join

Keyword: `\theta_join`

```
Question \theta_join_{Question.id = Answer.id} Answer;
```

Theta join evaluates each tuple in the Cartesian product of two relations against a boolean expression passed in as a required argument. The boolean expression can be composed of attribute references, numbers, quoted strings, comparison operators or logical operators. Attribute references must be valid.

Set operators

The right-hand and left-hand sides of the operator must be valid RA expressions. The attribute names of the relations that the expressions evaluate to must match exactly in names and order.

Union

Keyword: `\union`

```
Question \union Question2;
```

Difference

Keyword: `\difference`

```
Question \difference Question2;
```

Intersection

Keyword: `\intersect`

```
Question \intersect Question2;
```