SPJRUD to SQL Converter

Introduction

This library allows you to write database queries and execute them in a SQLite database. The SPJRUD expressions are first checked according to a database pattern and can then be translated to SQL.

Please note that NULL values are not supported.

This software is written as part of a project for the course of Databases I, given by Professor Jef Wijsen at the University of Mons (UMONS).

Usage

To use the module, you must first create a `SQLiteDB` object. This is the interface between the SQLite database and the SPJRUD to SQL Converter. To open execute queries using the database stored in the `example.db` file:

```
from algebra import *
db = SQLiteDB("example.db")
```

Once your have a database connector, you can start performing queries, either as `str` or `Expression` by using the `execute()` command. For example:

```
db.execute("SELECT * FROM my_table")
db.execute(Relation("my_table")["id"])
```

Please note that SQL queries sent by text are not checked in any way before executing them on the SQLite connector, whereas SPJRUD are checked before execution.

To save the changes if you made changes to the database, use `commit()`:

```
db.commit()
```

SPJRUD expressions

Selection

The selection is separated in two operations: `SelectionConstant` and `SelectionAttribute`.

- `SelectionConstant(attribute, constant, expression)`

Selects all rows of the expression where the value of the attribute mentioned has the value of the

constant. `attribute` can be either an Attribute object or a str with the name of the column to select. `constant` can be either a Constant object or a str/int/float with the desired value. `expression` is the table on which the selection is done.

- `SelectionAttribute(attribute1, attribute2, expression)`

Selects all rows of the expression where the value of the two attributes are the same. `attribute1` and `attribute2` can be either an Attribute object or a str with the name of the column to select. `expression` is the table on which the selection is done.

Projection

`Projection(list_of_attributes, expression)`

Reduces the attributes of the table to the ones mentioned. `list_of_attributes` is the list of the attributes as str to keep in the table. `expression` is the table on which the projection is done.

The shortcut `Expression[list_of_attributes]` is also available.

For example, the two following queries are similar:

```
q1 = Project(["id","name"], Relation("users"))
q2 = Relation("users")["id", "name"]
```

Join

`Join(expression1, expression2)`

Merges to tables together. `expression1` and `expression2` are the tables to be merged.

The shortcut `Expression1 * Expression2` is also available.

For example, the two following queries are similar:

```
q1 = Join(Relation("students"), Relation("marks"))
q2 = Relation("students") * Relation("marks")
```

Rename

`Rename(from, to, expression)`

Changes the name of one attribute. `from` is the attribute to change the name. `to` is the new name of the attribute. `expression` is the table on with the attribute will be renamed.

`from` and `to` can be Attribute objects, or str.

Union

`Union(expression1, expression2)`

Returns tuples from the 2 expressions which have the same attributes. `expression1` and `expression2` are the 2 tables on which the union is done.

The shortcut `Expression1 + Expression2` is also available.

For example, the two following queries are similar:

```
s1 = SelectConstant("name", "Alice", Relation("users"))
s2 = SelectConstant("name", "Bob", Relation("users"))

q1 = Union(s1, s2)
q2 = s1 + s2
```

Difference

`Difference(expression1, expression2)`

Returns tuples from expression1 that are not in expression2 where expression1 and expression2 must have the same attributes. `expression1` and `expression2` are the 2 tables on which the difference is done.

The shortcut `Expression1 - Expression2` is also available.

For example, the two following queries are similar:

```
s = SelectConstant("name", "Alice", Relation("users"))

q1 = Difference(Relation("users"), s)
q2 = Relation("users") - s
```