

# Violet (VPL) Database Manual

Don Batory  
 batory@cs.utexas.edu  
 May 2017

## 1 VIOLET

Violet is a free Java tool for drawing UML class diagrams. Familiarize yourself with Violet by installing MDELite and invoking it:

```
> java MDL.Violet
```

and drawing the diagram below:

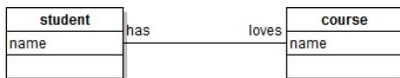


Fig. 1: Student-Course Diagram.

Draw your own diagrams. When you feel comfortable, proceed to the next section.

### 1.1 Violet Database Schema

Violet encodes a class diagram as an ugly XML file. The `MDL.ClassVioletParser` translates Violet-XML documents into a database that conforms to the following schema:<sup>1</sup>

```

dbase (vpl, [violetClass, violetInterface,
             violetAssociation, violetMiddleLabels]).

table (violetClass, [id, "name", "fields", "methods", x, y]).
table (violetInterface, [id, "name", "methods", x, y]).
table (violetAssociation, [id, "role1", "arrow1", type1,
                          "role2", "arrow2", type2, "bentStyle", "lineStyle",
                          cid1, cid2]).
table (violetMiddleLabels, [id, cid1, cid2, "label"]).
  
```

Some notes:

- `violetClass` is the name of a table with 6 columns.
  - `id` is the internal identifier of a class.
  - `x, y` are coordinate positions at which this class is displayed.
  - `name, fields, methods` are single-quoted strings which contain the name of the class, the string of fields and methods for that class.
- `violetInterface` is the name of a table with 5 columns.
  - `id` is the internal identifier of a class.
  - `x, y` are coordinate positions at which this class is displayed.
  - `name, fields, methods` are single-quoted strings which contain the name of the class, the string of fields and methods for that class.
- `violetAssociation` is a table with 11 columns.
  - `id` is the internal identifier of an association.
  - `cid1, cid2` are identifiers of classes or interfaces to be connected.
  - `type1, type2` are enums with values `classnode` or `interfacenode` to type `cid1` and `cid2`.
  - `role1, role2` – is any text *eg*, name and/or cardinality to be displayed.
  - `arrow1, arrow2` – only values {`V`, `NONE`, `TRIANGLE`, `DIAMOND`, `BLACK_DIAMOND`}, where `NONE` is ''.
  - `linestyle` – only values {`SOLID`, `DOTTED`}.
  - `bentStyle` – only values {`STRAIGHT` (or `blank`), `HV`, `VH`, `HVH`, `VHV`}.
- `violetMiddleLabels` is a table of 5 columns. The translation of violet to vpl assumes that all middle labels of associations are empty. For each non-empty middle label, a tuple appears in this table, along with the `cids` of the classes/interfaces that are connected. Basically this table should never have rows. If it has tuples, flag an error!
  - `id` is the internal identifier of an association.
  - `cid1, cid2` are identifiers of classes or interfaces to be connected.
  - `label` is the middle label of the association.

Let `SC.class.violet` be the Violet-produced XML file for Figure 1. The following MDELite command translates this file into a VPL database `SC.vpl.pl`:

```
> java MDL.VioletClassParser SC.class.violet SC.vpl.pl
```

<sup>1</sup>I have broken lines in code listings for presentation reasons. MDELite parsers expect one complete declaration per line.

The database that is produced is:

```

dbase(vpl, [violetMiddleLabels, violetAssociation,
           violetInterface, violetClass]).

table(violetClass, [id, "name", "fields", "methods", x, y]).
violetClass(classnode0, 'student', 'name', '', 335, 133).
violetClass(classnode1, 'course', 'name', '', 618, 127).

table(violetInterface, [id, "name", "methods", x, y]).

table(violetAssociation, [id, "role1", "arrow1", type1,
                          "role2", "arrow2", type2, "bentStyle", "lineStyle",
                          cid1, cid2]).
violetAssociation(id0, 'has', '', classnode, 'loves', '',
                 classnode, '', '', classnode0, classnode1).

table(violetMiddleLabels, [id, cid1, cid2, "label"]).

```

## 1.2 VPL Constraints

There indeed are VPL constraints. I have not posted them, as they are good examples for homework assignments. They are similar to those for YPL (VPL's Yuml counterpart).

## 2 CLOSING

MDELite is a work in progress. It is possible that this documentation may get out-of-date with code releases. If so, please report them to me — dsb