Technical aspects of VTL to SQL translation

Marek Panfiłow
M.Panfilow@stat.gov.pl
VtlProcessingLib

Initial functionality

- Generation of output code of target programming language (initially T-SQL)
- Optimization and type checking of input VTL compilation unit
- Generation of compiled VTL modules from their source codes to be persistently stored for later reuse
- Analysis of input VTL program or module for list of compile-time errors
- Modular architecture that allows extending functionality
VtlProcessingLib

Modular architecture that allows extending functionality by adding new:

• Supported VTL versions and dialects
• Individual standard library operators, functions or procedures
• Supported target languages (e.g. SAS, R, C#)
• Components for optimizing and transforming VTL compilation units
VtlProcessingLib

Possible usage

• Stand-alone applications managing and running VTL programs
• Development environments for defining VTL rules
• Shared services for validating and running VTL programs
Technical architecture

• FRONT END – verifies syntax and semantics according to a specific source language. It includes lexical analysis, syntax analysis and semantic analysis stages. Eventually generates an intermediate representation (abbreviated as IR) of the source code for processing by the middle-end. This IR is usually a lower level representation of the program with respect to the source code.

• MIDDLE END – performs transformations and optimizations on intermediate representation of the source code. This allows for transformations which are independent to both source and target languages syntax. Middle-end produces modified/optimized IR for further processing.

• BACK END – takes the output from the middle-end, then generates the target language code. Some more optimizations specific to target language might be done at this stage.
Technical architecture

- Different front-end implementations are needed for supporting different dialects of VTL
- Intermediate Compilation Unit - intermediate representation
- Persistent object mapping
- Supported target languages (e.g. SAS, R, C#)
- The target language specific part is supplied to the back-end as replaceable modules.
Technical architecture

- **HEADER** – It contains all the information about compilation unit itself. That consists of a version of VTL syntax, a module name and a list of dependencies. Some more compilation unit metadata may be added at later stages of development.

- **DEFINITIONS** – This is a collection of persistent artefacts of VTL information model defined in compilation unit source code with VTL model definition language. It is primarily used by VTL “get” operator to acquire input data structures.

- **MANIPULATIONS** – This is information about data manipulations performed by compilation unit. It contains a collection of graph structures, each of which corresponds to a single VLT function, procedure or transformation. These graphs are the objects for optimizations and transformations mostly done in the middle-end phase of translation.
Implementation details
Proof of concept software
Proof of concept software
Proof of concept software

```sql
1  ds1 := get("dbo.data0001")
2  ds2 := get("dbo.test1")
```

```sql
-- VTL: ds1 := get("dbo.data0001")
SELECT [K1], [K2], [K3], [M1], [M2], [M3] INTO #c
-- VTL: ds2 := get("dbo.test1")
SELECT [n], [v], [p] INTO #ds2 FROM dbo.test1
```
Proof of concept software
Planned dates for the regional conferences:
Lisbon    13.11.2017 – 14.11.2017
Den Haag  11.01.2018 – 12.01.2018
Vilnius   01.02.2018 – 02.02.2018
Belgrade  22.02.2018 – 23.02.2018
Summary and final remarks

• VTL 1.1 specification was not yet released.
• Aggressive inlining strategy

➢ There is still a lot to be done