ISBN 978-981-11-3671-9
Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)
Beijing, 25-27 June, 2017, pp. 219-223
doi: 10.18178/wcse.2017.06.038

Case Study of ETL Java Code Generation from Domain Specific Language

Sunisa Junsawang and Yachai Limpiyakorn [+]

Department of Computer Engineering, Chulalongkorn University, Bangkok 10330, Thailand

Abstract. Extract-Transform-Load, ETL, is the integration layer in data warehouse environment. The quality of ETL contributes to the accuracy and the correctness of data, it is expensive, though. This paper aims to improve the process productivity and usability. A domain-specific language has been developed for scripting ETL processes. Alternatively, the input of ETL scripts written by non-technical users in Excel format is also allowed. The paradigm of code-to-code transformation is applied for transforming the textual, domain-specific language into Java code. The etlDSL code generator is implemented with MVC architecture design. A case study and the user evaluation of the proposed approach are also discussed. The overall rating is satisfaction.

Keywords: domain specific language, ETL, query, data warehouse, software process improvement.

1. Introduction

In data warehouse environment, Extract-Transform-Load (ETL) processes constitute the integration layer responsible for pulling data from data sources to targets, via a set of transformations. ETL is a critical component contributes to the accuracy and the correctness of data, which are key factors of the success or failure of data warehouse projects. However, building ETL processes consumes up to 70% of resources regarding time, money and effort [1]. To improve the process productivity and usability, we develop a domain-specific language (DSL) for scripting the ETL process. The prototyping code generator, etlDSL, is implemented to support separation of concerns by means of Model-View-Controller (MVC) architecture design as described in the earlier work [2]. In this paper, the Interface layer of the View component is enhanced with the provision of Excel templates for user-defined ETL scripts to promote the usability. A case study and the user evaluation of the proposed approach are also discussed.

In literature, Hemel et al. [3] presented a case study of the code generation by model transformation approach applied in the development of WebDSL, which is a domain-specific language for dynamic web applications. The WebDSL generator follows the four-level model organization to transform high-level models into Java code and XML files. Oliveira and Belo [4] proposed the use of Business Process Modelling Notation (BPMN) for building the simulator of ETL process. The DSL grammars were defined and translated to Java language by Xtext tool. Deneke [5] developed the DSL by C# for building ETL workflows. The Object-Relational Mapper (ORM) was used for data gathering instead of data query with SQL. Lasya and Tanuku [6] proposed an approach to optimizing the executable code generated from a certain SQL query. Several alternative query trees are constructed based on the relational algebra expressions underlying the input SQL query. The Query Execution Plan associated with the query tree that provides the minimum cost will be selected for executable code generation. Santos and Belo [7] proposed modelling ETL conciliation tasks using relational algebra operators. Several scenarios of ETL were illustrated in relational algebra trees for integrity awareness.

[+] Corresponding author. Tel.: + 668 2218 6959; fax: +668 2218 6955.
E-mail address: *Yachai.L@chula.ac.th*.

2. etlDSL

The DSL commands contain the common vocabularies familiar to the domain experts. In earlier work [2], we described the development of a domain-specific language created for scripting ETL processes. The current DSL presented in this paper has been slightly improved. Fig. 1 illustrates its meta-model which is a grammar describing the valid sentences of the language. The etlDSL code generator is also implemented providing alternative input of ETL scripts written by non-technical users, such as Business Analyst (BA). The overview of the proposed ETL process is depicted in Fig. 2.

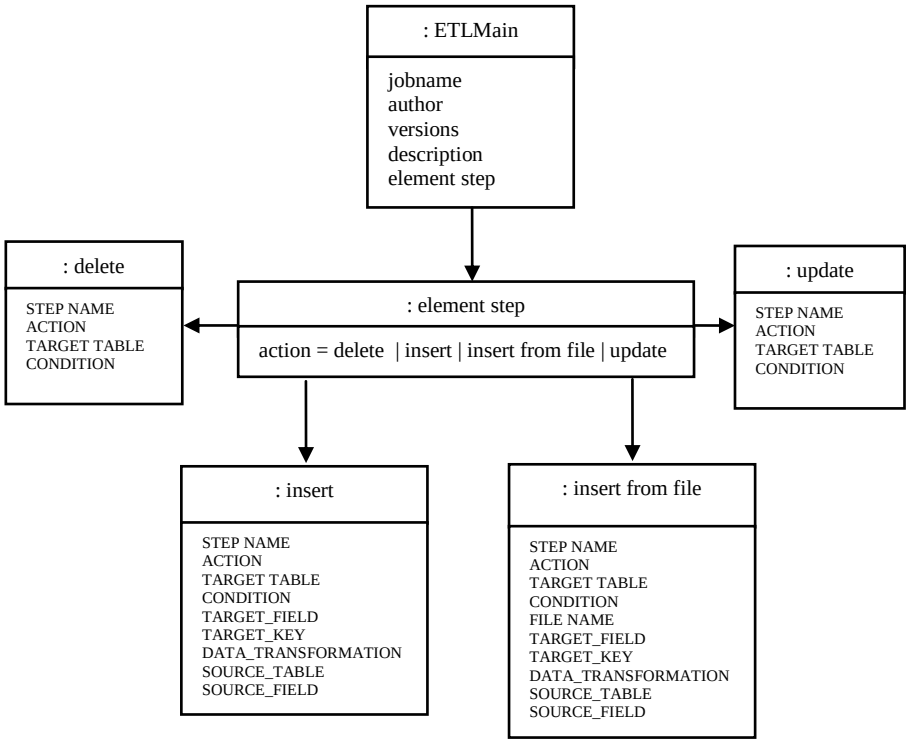

Fig. 1: DSL Meta-model.

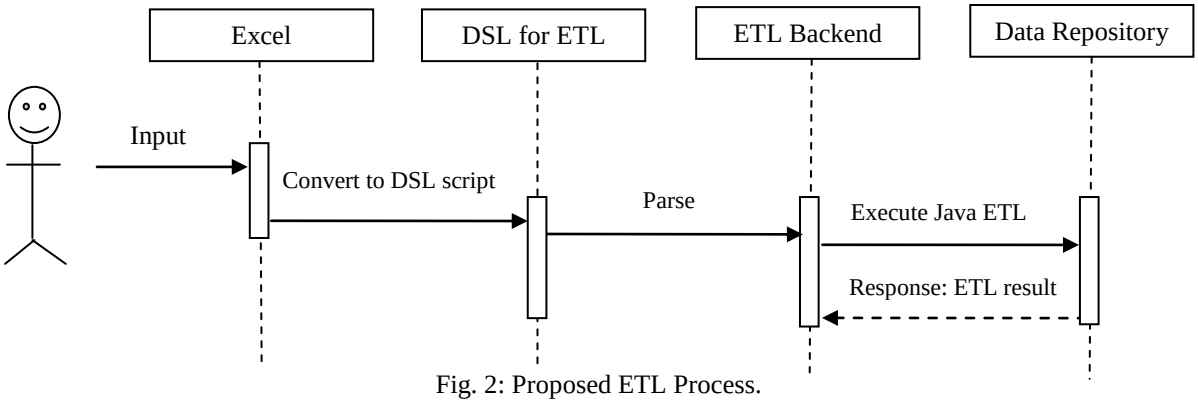

Fig. 2: Proposed ETL Process.

Non-technical users can traditionally describe ETL processes in Excel files and then save to .etl extension, i.e. automatically convert to DSL script. Alternatively, they can directly write the DSL scripts. As shown in Fig. 3, the example ETL script contains Action *delete* and *insert* into the target table CAMPAIGN_FACT with the updates. The data transformation is categorized into four types:

1) Constraint: e.g. move direct, null.
2) Function: e.g. nvl(), sum().
3) Number: e.g. 0, 1.
4) String: e.g. ‘USE’.

STEP NAME	Delete CAMPAIGN
ACTION	delete
TARGET TABLE	CAMPAIGN_FACT
CONDITION	(STORE,DT_KEY) in (select l.STORE,l.DT_KEY from campaign_fct c, STG_LOG l where c.STORE = l.STORE and c.DT_KEY = l.DT_KEY)
STEP NAME	CAMPAIGN use_cards
ACTION	insert
TARGET TABLE	CAMPAIGN_FACT (update and insert)
CONDITION	sale_attribute.saleheaderid = sales_header_fct.sale_id and sales_header_fct.store = store_dim.store and sale_attribute.att056 = '2' group by sale_attribute.saleheaderid, to_char(to_date(substr(sale_attribute.att040, 1, 10),'yyyy-mm-dd'),'yyyymmdd'), sales_header_fct.dt_key, sales_header_fct.store, store_dim.store_nm, sales_header_fct.channel_sr_key, sale_attribute.att039,sale_attribute.ATT054 ,sale_attribute.ATT055 , sale_attribute.ATT057)

TARGET_FIELD	TARGET_KEY	DATA_TRANSFORMATION	SOURCE_TABLE	SOURCE_FIELD
DT _KEY		move direct	sales_header_fct	dt_key
CHANNEL_SR_KEY		move direct	sales_header_fct	channel_sr_key
STORE	key	move direct	sales_header_fct	store
STORE_NM		move direct	store_dim	store_nm
CARD_NUMBER	key	move direct	sale_attribute	att046
SALE_CARD_SALEID	key	move direct	sale_attribute	saleheaderid
USE_CARD_SALEID		null		
SALE_CARD_QTY		0		
USE_CARD_QTY		nvl(sum(att043),0)		
TRUN_TYPE		'USE'		
DATA_LOAD_DT		sysdate		
DATA_UPDT_DT		sysdate		
ATTR		move direct	sale_attribute	ATT054
ATTR2		move direct	sale_attribute	ATT055
ATTR3		move direct	sale_attribute	ATT057
ATTR4		' '		
ATTR5		' '		
NUM_SALE_CARD_QTY		0		
NUM_USE_CARD_QTY		1		

Fig. 3: Example ETL Script in Excel File.

Next, Xtext, an open-source framework for developing programming languages and domain-specific languages, is used as the parser generator. The DSL script will be parsed to Java code. Fig. 4 illustrates the DSL script associated with the ETL Excel script shown in Fig. 3. The execution of Java code will then invoke the ETL backend to create SQL statements using a set of predefined transformation rules, examples as shown in Fig. 5. The generation of SQL statements in this work supports full outer join, inner join, left join, and right join. Finally, the target model will be created by merging all the created target meta-models with the predefined templates of standard Java commands and the log reporting the execution status.

```
jobname: InsetDataIntoCAMPAING_FACT
author: Sunisa J
version: v 1
description: This is the test script for testing DSL on ETL tool

STEP NAME|Delete CAMPAING|
ACTION|delete|
TARGET TABLE|CAMPAIGN_FACT|
CONDITION|"(STORE,DT_KEY) in
(select l.STORE,l.DT_KEY
from campaign_fct c, STG_LOG l
where c.STORE = l.STORE
and c.DT_KEY = l.DT_KEY)"|

STEP NAME|CAMPAING use_cards|
ACTION|insert|
TARGET TABLE|CAMPAIGN_FACT (update and insert)|
CONDITION|"sale_attribute.saleheaderid = sales_header_fct.sale_id
and sales_header_fct.store = store_dim.store
and sale_attribute.att056 = '2'
group by sale_attribute.saleheaderid,
to_char(to_date(substr(sale_attribute.att040, 1, 10),'yyyy-mm-dd'),'yyyymmdd'),
sales_header_fct.dt_key,
sales_header_fct.store,
store_dim.store_nm,
sales_header_fct.channel_sr_key,
sale_attribute.att039,sale_attribute.ATT054,
sale_attribute.ATT055,sale_attribute.ATT057)"|
COLUMNS|TARGET_FIELD|TARGET_KEY|DATA_TRANSFORMATION|SOURCE_TABLE|SOURCE_FIELD
VALUES|DT_KEY||move direct|sales_header_fct|dt_key
VALUES|CHANNEL_SR_KEY||move direct|sales_header_fct|channel_sr_key
VALUES|STORE|key|move direct|sales_header_fct|store
VALUES|STORE_NM||move direct|store_dim|store_nm
VALUES|CARD_NUMBER|key|move direct|sale_attribute|att046
VALUES|SALE_CARD_SALEID|Key|move direct|sale_attribute|saleheaderid
VALUES|USE_CARD_SALEID||||
VALUES|SALE_CARD_QTY||0||
VALUES|USE_CARD_QTY||nvl(sum(att043),0)||
VALUES|TRUN_TYPE||###USE###||
VALUES|DATA_LOAD_DT||sysdate||
VALUES|DATA_UPDT_DT||sysdate||
VALUES|ATTR||move direct|sale_attribute|ATT054
VALUES|ATTR2||move direct|sale_attribute|ATT055
VALUES|ATTR3||move direct|sale_attribute|ATT057
VALUES|ATTR4||### ###||
VALUES|ATTR5||### ###||
VALUES|NUM_SALE_CARD_QTY||0||
VALUES|NUM_USE_CARD_QTY||1||
```

Fig. 4: DSL Script Associated with ETL Excel File in Fig. 3.

Rule1: Delete

1.Get parameters from DSL (type, table name, condition)

2.*If* type equals 'clear' *then* truncate table target *else* get target table name and condition to generate SQL *delete* statement

3.Execute statement of (2)

Rule3: Update from database/ input value(s)

1.Get parameters from DSL (type, table name, condition)

2.Load mapping values to mapping table

3.Create SQL *merge* statement from mapping table

If type equals 'update and insert' *then* generate 'when not match then insert' *else* does not generate 'when not match then insert'

4. Execute statement (3)

5. Clear data in mapping table

Rule2: Insert from file

1. Get parameters from DSL (type, table name, file name, mapping values)

2. Load mapping values to mapping table

3. Create temporary table based on column of file

4. Load file to temporary table

5.Generate SQL statement from data in mapping table

6. *If* type equals 'clear and insert' *then* delete data in target table and generate SQL *insert* statement on data in mapping table *else* generate SQL *insert* statement on data in mapping table

7.Execute statement of (6)

8.Drop temporary table and clear data in mapping table

Fig. 5: Example SQL Transformation Rules.

3. Evaluation

The satisfaction survey was conducted based on the quality dimensions including: Operability (users' efforts for operation and operation control), Learnability (how well users can learn the application of

software), Correctness (if the ETL script functions correctly), Fitness for Use (if the system fulfills intended use), and Overall Satisfaction. Six people who are project manager, business analysts, tester, programmer, and QA were selected for writing the DSL script (Fig. 4). Table 1 summarizes the results reporting the overall averaging score equals to 4.07 or achieving satisfaction ranking.

Table 1: Satisfaction Survey Results.

	Score					Average Score
	5	**4**	**3**	**2**	**1**	
Operability	4	1	1	0	0	4.5
Learnability	1	4	1	0	0	4
Correctness	6	0	0	0	0	5
Fitness for Use	0	1	5	0	0	3.17
Overall Satisfaction	0	4	2	0	0	3.67
Overall Average						**4.07**

Notes: Ranks: 1-very unsatisfaction; 2-unsatisfaction; 3-neutral; 4-satisfaction; 5-very satisfaction.

4. Conclusion

To support ETL layer in data warehouse environment, this paper presents the development of etlDSL generator to transform high-level abstractions into Java code. With Xtext framework, the proposed approach can be also applied for the transformation to other languages, such as C#, .NET. This will require the modification of ETL Backend to support word truncation in other programming languages. A case study of ETL written in DSL is discussed and evaluated. The results showed that the users are satisfied with the proposed approach. The etlDSL system could function the ETL process correctly. The users can learn and operate the application software with not much effort. However, some user feedbacks suggest more pilot studies on existing ETL processes should be conducted before widely used.

5. References

[1] A. Kabiri and D. Chiadmi, "SURVEY ON ETL PROCESSES," *Journal of Theoretical and Applied Information Technology*, vol. 54, 2013.D. Ghosh, *DSLs in Action*: Manning Publications, 2011.

[2] S. Junsawang and Y. Limpiyakorn, "A Domain Specific Language for Scripting ETL Process," 2017.

[3] Z. Hemel, L. C. L. Kats, and E. Visser, "Code Generation by Model Transformation: A Case Study in Transformation Modularity," 2008.

[4] B. Oliveira and O. Belo, "On the specification of extract, transform, and load patterns behavior: A domain‐specific language approach," 2016.

[5] W. Deneke, "A Domain Specific Model for Generating ETL Workflows from Business Intents," Doctor of Philosophy in Computer Science, University of Arkansas, Fayetteville, University of Arkansas, Fayetteville ScholarWorks@UARK, 2012.

[6] S. Lasya and S. Tanuku, "A Study of Library Databases by Translating Those SQL Queries Into Relational Algebra and Generating Query Trees," 2011.

[7] V. Santos and O. Belo, "Modelling ETL Conciliation Tasks Using Relational Algebra Operators," presented at the 2014 UKSim-AMSS 8th European Modelling Symposium, 2014.