Towards Platform Independent Database Modelling in Enterprise Systems

Martyn Ellison, Radu Calinescu, and Richard F. Paige

Department of Computer Science
University of York, UK
{mhe504, radu.calinescu, richard.paige}@york.ac.uk

Abstract. Enterprise software systems are prevalent in many organisations, typically they are data-intensive and manage customer, sales, or other important data. When an enterprise system needs to be modernised or migrated (e.g. to the cloud) it is necessary to understand the structure of this data and how it is used. We have developed a tool-supported approach to model database structure, query patterns, and growth patterns. Compared to existing work, our tool offers increased system support and extensibility which is vital for use in industry. Standardisation and platform independence is ensured by producing models conforming to the Knowledge Discovery Metamodel and Software Metrics Metamodel.

1 Introduction

Model driven engineering has been shown to aid the modernisation and re-engineering of enterprise software systems [18]. Public clouds are a common target platform for these systems, as investigated in the ARTIST [9] project and CloudMIG [6]. However, this existing work gives minimal consideration to data (and the database layer) despite this being the most valuable and irreplaceable part of many systems [5]. Extensive work has been done on database modelling [2] [12] and reverse engineering [16] [4], although this is disconnected from the work on software modernisation and does not focus on the cloud.

In order to determine the costs of migrating and storing data in the cloud the workload of the enterprise system's database, i.e., query patterns and growth patterns, must be known. Furthermore, the database structure must be known to decide which tables or columns to migrate and whether any modernisation tasks should be performed during migration (e.g., table merging). In this paper we investigate how to model these properties in a platform independent way so that further analysis and migration simulation is possible.

We have developed a prototype model extraction tool, called DBLModeller, which transforms a schema dump and query log into a structure and a workload model. These conform to the Knowledge Discovery Metamodel (KDM) and the Structured Metrics Metamodel (SMM) respectively. Most existing model-driven modernisation and cloud migration approaches use these metamodels, and the range of existing tools mean they are an ideal choice for our work. Approaches for obtaining these two inputs are also proposed.

DBLModeller has been developed in partnership with Science Warehouse (`www.sci-ware.com`), a UK-based e-business company that specialises in enterprise procurement software. They have provided access to their core system to support the evaluation of DBLModeller. This is a large Java-based system that business customers use to order and compare products from multiple suppliers.

One key challenge when modelling databases is heterogeneity, as a variety of SQL dialects exist. This becomes an issue when developing a platform independent tool which supports a wide range of systems. We have overcome this in a number of ways: (1) our approach incorporates existing tools from database providers, (2) a single grammar has been developed to support multiple SQL dialects, and (3) a model-to-model transformation has been removed from the model extraction process required to produce KDM and SMM models. These changes increased the range of enterprise systems supported by DBLModeller and improved extensibility compared to the state-of-the-art Gra2MoL SQL-to-KDM tool [8].

The rest of the paper is structured as follows. Section 2 presents our DBLModeller approach. Section 3 describes our evaluation of DBLModeller. Section 4 concludes the paper with a brief summary and suggests future work directions.

2 DBLModeller Approach

Our approach for producing the structure (KDM) and workload (SMM) models is shown in Figure 1. Step 1 (T1 & T2) is performed by existing external tools, such as Oracle SQL Developer [15] for the schema and P6Spy [1] for the SQL trace. Nearly all databases will have an accompanying management tool which can produce a SQL schema dump, this is preferable to using any existing SQL scripts in the organisation as these may be out of date. Similarly, several approaches exist to capture a SQL trace (i.e., the SQL queries being sent to the database) but we propose using a 'spying' library. These wrap the existing database driver used by the system and save all queries to a log file, therefore code changes to the system are not required.

DBLModeller can process the SQL traces produced by P6Spy into a sequence of load measurements (Step 1 T3), if the SQL trace is captured via another method then this must be done manually. The load measurements include: entity count, entity reads, entity writes, unused entities, and database size. An 'entity' is the primary data artefact the database is storing and is selected by the user; e.g. products in an e-commerce system or pages in a Wiki. Abstracting the measurements in this way greatly simplifies the workload model extraction, and makes the process more generic.

Steps 2 & 3 are fully automated and performed by DBLModeller. The text-to-model transformation (Step 2) consists of two separate transformations, one

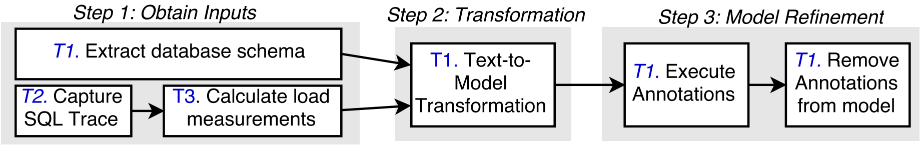
Step 1: Obtain Inputs
T1. Extract database schema
T2. Capture SQL Trace
T3. Calculate load measurements
Step 2: Transformation
T1. Text-to-Model Transformation
Step 3: Model Refinement
T1. Execute Annotations
T1. Remove Annotations from model

Fig. 1. Overview of the DBLModeller approach

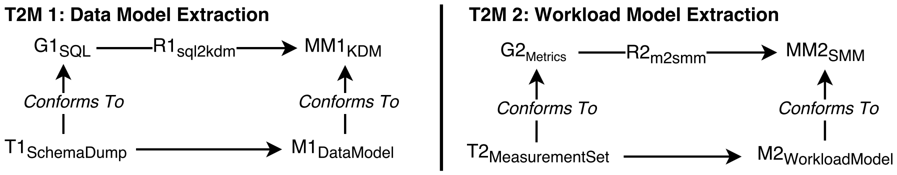

Fig. 2. Text-to-model transformations in DBLModeller

for each metamodel, as shown in Figure 2. This uses grammar-to-model mapping (the approach from [7]), with SQL and Metrics grammars (G1 & G2) that are mapped to the KDM and SMM (MM1 & MM2) via a set of rules we developed.

Traditionally when using a grammar-to-model approach a publicly available grammar is reused, however for DBLModeller we developed our own multi-dialect SQL grammar. Given that the source grammar and mapping rules are tightly coupled, a new rule-set (i.e, R1 on Figure 2) would be required for each grammar. This would not be a scalable approach if a large number of SQL dialects were to be supported. A multi-dialect grammar is feasible when the target metamodel (KDM) is at a far higher level of abstraction than the source language, e.g., the TEXT (in MySQL) and VARCHAR (in Oracle) types both map to the KDM type String. Furthermore, support for a new dialect can be achieved by adding the unsupported constructs to the grammar (many constructs will exist in both) and without modifying the mapping rules, improving extensibility.

If a model is highly abstract or significantly different from a source text, model extraction typically requires a model-to-model (M2M) transformation to be combined with a text-to-model (T2M) transformation [8] [13]. This is undesirable when designing a model extraction tool to be extensible, because the M2M transformation is an additional code block to modify. In the case of KDM model extraction, the M2M transformation is only needed to perform simple tasks like moving elements or resolving references. We developed an annotation-based model-extraction approach to remove simple M2M transformations.

In our approach three annotation types exist: Add, Move, and Reference. These are inserted during the T2M transformation (Step 2) to produce an annotated model, which is processed by a model refinement engine (Step 3). The user will never see the annotated model and he/she will never have to consider which annotations to use unless DBLModeller is being extended to support a new SQL dialect or version. We have used annotations to support: ALTER statements, CREATE INDEX statements, and to resolve references. These three use cases require the model to be searched after the initial T2M transformation, e.g. an ALTER statement might add a primary key to a previously created table, this model element must be found and modified.

3 Evaluation

The evaluation of DBLModeller is based around the following research questions:

RQ1 To what extent are the models produced by DBLModeller complete and correct?

RQ2 How is the model extraction time impacted by the M2M transformation replacement?

RQ3 How does the set of SQL keywords supported by DBLModeller compare to those used in Oracle/MySQL dumps from real-world systems?

RQ4 What is the effect of a multi-dialect grammar and annotated T2M transformation on extensibility?

3.1 Model Extraction

This section evaluates the completeness and correctness of the extracted models (RQ1) and the performance of DBLModeller (RQ2). These research questions have been examined together as they impact on each other. DBLModeller has been compared to Gra2MoL's PLSQL2KDM example [3] as this had the highest level of SQL support at the time of writing. We extracted KDM models from the database schemas of four systems: Apache OFBiz, MediaWiki, Science Warehouse, and a student record system [8]. With OFBiz and MediaWiki we obtained Oracle and MySQL versions of the schema by installing them on both databases. Additionally, SMM models were extracted from Wikipedia (using 6 months data from [10] and [19]) and Science Warehouse.

Model completeness was assessed by comparing the number of model elements and input elements, while for correctness the properties of the model elements and input elements were compared. We developed a small model checking tool to automate this analysis. DBLModeller was able to extract models from the 6 schemas successfully, and from the output of our tool we concluded: that the input text and the output model had the same number of elements, all table, column, and sequence names were correct, and relationships between tables were correct. Furthermore, we confirmed the models conformed to KDM and SMM using the Eclipse Modelling Framework.

The performance of DBLModeller was assessed by extracting a KDM model for each schema and measuring the time taken. This process was repeated 20 times per schema. We expected that the removal of the M2M transformation from the model extraction process will have significant performance gains. A virtual machine on the Digital Ocean cloud platform with 4GB of RAM and two Ivy Bridge based Intel Xeon cores was used to perform the experiment.

The performance results are presented in Table 1, which shows that DBLModeller can extract a KDM model in less time for every schema. As Gra2MoL

Table 1. Model Extraction Times using DBLModeller and Gra2MoL

Schema	Size (KLOC)	Tool	Mean (Secs.)	Std. Dev.	sec/KLOC
Oracle OFBiz	31.5	DBLM	174	2.35	6
	10.3	G2M	237	3.4	24
Oracle MediaWiki	2	DBLM	7	0.23	4
	0.8	G2M	14	0.68	18
Oracle Science Warehouse	1	DBLM	5	0.19	5
	0.4	G2M	14	0.62	35
Oracle UoM Student System	0.3	DBLM	5	0.21	17
	0.3	G2M	10	0.62	33
MySQL OFBiz	21.7	DBLM	104	2.13	5
	9.5	G2M	230	9.73	24
MySQL MediaWiki	1	DBLM	5	0.24	5
	0.4	G2M	13	0.53	33

supports fewer SQL statements than DBLModeller, in order to obtain results it was necessary to modify the schemas by removing unsupported content until they could be processed by Gra2MoL. We used the metric "sec / KLOC" to simplify comparison, based on this we conclude the model extraction time has been reduced by up to 86% for Oracle schemas and up to 84% for MySQL.

3.2 RQ3: SQL Keyword Usage Study

Fully supporting every SQL dialect was impractical due to the number that exist and the size of the language, therefore DBLModeller supports a subset of two dialects (Oracle and MySQL). Whilst it is straightforward to identify which dialects to support (many organisations report on the estimated market share), it is harder to select statements and keywords to support within these. We have re-used the 6 schemas from Section 3.1 and obtained 9 others, giving a sample size of 15 (listed in Table 2). The additional schemas were obtained using the same process, i.e., deploying an instance of the system then connecting to its database with MySQL Workbench or Oracle SQL Developer.

Figure 3 shows the 25 most used keywords in our schema set. None of the words which appear in the MySQL top 25 are unsupported, while only two in the Oracle top 25 are unsupported: NVARCHAR2 and USING. NVARCHAR2 is only used by Confluence (albeit extensively), however this means the data type for NVARCHAR2 columns will be null. If necessary, support for NVARCHAR2 could be added by modifying one line of the SQL grammar (to map it to the KDM:String data type). The lack of support for USING is not an issue because it specifies whether an index is enabled or disabled in the Confluence schema, and this detail is lost when abstracting to a KDM model. From the keyword anyalsis data we can also conclude that DBLModeller supported 96% of the content in the MySQL schemas and 99% of the content in the Oracle Schemas.

Table 2. Database schemas used for keyword analysis

System	Type	Domain
Science Warehouse	Oracle	E-commerce
Record System [8]	Oracle	Record System
Apache OFBiz	Oracle & MySQL	Business Management & E-commerce
MediaWiki	Oracle & MySQL	Collaboration
Confluence	Oracle & MySQL	Collaboration
Joomla	MySQL	Website Management
Magneto	MySQL	E-commerce
SonarQube	MySQL	Software Engineering
Mantis	MySQL	Software Engineering
WordPress	MySQL	Website Management
Moodle	MySQL	Education
OrangeHRM	MySQL	Record System
SuiteCRM	MySQL	Business Management
RefBase	MySQL	Education
OpenMRS	MySQL	Record System

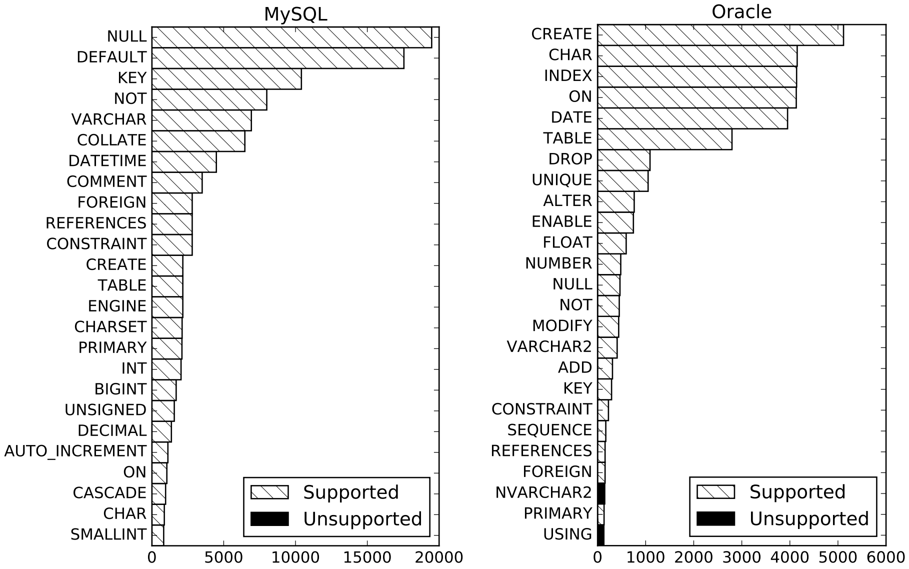

Fig. 3. Most frequent SQL keywords for the MySQL (left) and Oracle dialects (right)

3.3 RQ4: Microsoft SQL Server Specialisation

Given the heterogeneity of schemas, databases, and SQL dialects, it is inevitable that DBLModeller may need to be extended. A case study has been performed where DBLModeller and the Gra2MoL SQL-to-KDM extraction tool [8] [3] were extended to support a Microsoft SharePoint schema.

A schema dump was obtained from a Microsoft SQL Server database used by a Microsoft SharePoint 2013 instance. This instance was created specifically for the case study by installing SharePoint (the schema rather than the data is needed here, so the results are unaffected by the SharePoint instance not being live/in-use). As SharePoint uses 16 schemas the largest was selected; this contains 7 KLOC consisting of 136 tables, 5442 columns, and 61 indexes. The goal here was not to select a schema which is representative of all Microsoft SQL Server based databases, but rather to have a schema which is unsupported by both tools in equal measure.

The changes needed to DBLModeller to support the schema were determined by attempting to extract a model, then noting any errors produced. These were then fixed incrementally and the number of modified lines of code were counted. However, with Gra2MoL a new ANTLR grammar was developed to parse the schema dump. This makes it possible to compare the work required to extend a grammar against the work required to develop a new grammar.

Table 3 presents the results and shows that the extension of DBLModeller required fewer code changes. The use of our annotated T2M transformation meant that no M2M rule changes were needed. The use of a multi-dialect grammar meant it was unnecessary to write a new grammar for the Microsoft SQL dialect, instead we modified various rules in our existing grammar. However, when comparing the development time/effort in extending the two tools it is im-

Table 3. Code changes to support the SharePoint schema

	G2M New Grammar		**DBLModeller Extension**	
	New LOC	Updated LOC	New LOC	Updated LOC
Lexer	70	0	25	6
Parser	11	0	6	14
T2M	25	85	24	1
M2M	10	5	0	0
Total	*116*	*90*	*55*	*21*

portant to consider whether a new LOC represents the same effort in each. As identical technologies are used in the Gra2Mol PLSQL example and DBLModdler (ANTLR for the Lexer/Parser, and the G2M DSL for the T2M rules), the results should be comparable. Returning to RQ4, we conclude that the changes made in DBLModeller have had a positive effect on extensibility. Furthermore, the similarities between SQL dialects meant that extending the DBLModeller was a straightforward task.

4 Conclusions & Future Work

We introduced DBLModeller, a tool-supported approach for modelling the database layer of enterprise software systems. Using this we were able to extract a structure model conforming to the KDM [17] and a workload model conforming to the SMM [14]. These standardised metamodels ensured interoperability with exiting modelling and cloud migration tools [11] [9]. Previous database modelling tools did not capture the properties which influence cloud migration costs, i.e., growth and query patterns. Furthermore, we decoupled the extraction of KDM and SMM based models from their use (e.g. SMM was used within CloudMIG [6] but the user is not able to access the model and other SMM models can not be used as input).

We evaluated DBLModeller using database schemas and log files from multiple real systems. Our experiments showed that DBLModeller can extract models from a wider range of systems and can be extended with less effort than the leading existing tool (Gra2MoL [8]). These key benefits were achieved by removing a model-to-model transformation from the model extraction process and by using a single multi-dialect grammar instead of using a grammar for each dialect.

In the future we plan to use DBLModeller to extract models from other live systems and to use these to simulate and optimise data migration from legacy systems into the cloud. Other potential areas of future research include analyses of the extracted models to identify design issues or anti-patterns, and to identify suitable database types for the system being modelled. Finally, we envisage that the use of annotations to avoid the need for multiple model transformations will have applications in other areas of model-driven software engineering.

Acknowledgements

This work was funded by the UK EPSRC grant EP/F501374/1. Science Warehouse Ltd granted access to their systems for evaluation purposes and provided feedback on the industrial application of DBLModeller.

References

1. P6Spy Framework. `http://p6spy.github.io/p6spy`
2. Alalfi, M.H., Cordy, J.R., Dean, T.R.: SQL2XMI: Reverse engineering of UML-ER diagrams from relational database schemas. In: Proceedings of the 15th Working Conference on Reverse Engineering. pp. 187–191 (2008)
3. Canovas, J.: Gra2Mol: PLSQL2ASTM example project. `github.com/jlcanovas/gra2mol/tree/master/examples/Grammar2Model.examples.PLSQL2ASTMModel`
4. Davis, K.H., Alken, P.: Data reverse engineering: A historical survey. In: Seventh Working Conference on Reverse Engineering. pp. 70–78. IEEE (2000)
5. Díaz, O., Puente, G., Izquierdo, J.L.C., Molina, J.G.: Harvesting models from web 2.0 databases. Software & Systems Modeling 12(1), 15–34 (2013)
6. Frey, S., Hasselbring, W.: Model-based migration of legacy software systems into the cloud: The CloudMIG approach. Softwaretechnik-Trends 30(2) (2010)
7. Izquierdo, J.L.C., Molina, J.G.: A domain specific language for extracting models in software modernization. In: 5th European Conference on Model-driven Architecture Foundations and Applications (ECMDA-FA). pp. 82–97 (2009)
8. Izquierdo, J.L.C., Molina, J.G.: An architecture-driven modernization tool for calculating metrics. IEEE Software 27(4), 37–43 (2010)
9. Menychtas, A., Konstanteli, K., Alonso, J., Orue-Echevarria, L., Gorroñogoitia, J., Kousiouris, G., Santzaridou, C., Bruneliere, H., Pellens, B., Stuer, P., Strauß, O., Senkova, T., Varvarigou, T.A.: Software modernization and cloudification using the ARTIST migration methodology and framework. Scalable Computing: Practice and Experience 15(2), 131–152 (2014)
10. Mituzas, D.: Page view statistics for Wikimedia projects. `dumps.wikimedia.org/other/pagecounts-raw`
11. Mohagheghi, P., Sæther, T.: Software engineering challenges for migration to the service cloud paradigm: Ongoing work in the REMICS project. In: IEEE World Congress on Services (SERVICES). pp. 507–514 (2011)
12. Mori, M., Noughi, N., Cleve, A.: Mining SQL execution traces for data manipulation behavior recovery. In: Joint Proceedings of the CAiSE 2014 Forum and CAiSE 2014 Doctoral Consortium co-located with the 26th International Conference on Advanced Information Systems Engineering (CAiSE 2014), Thessaloniki, Greece, June 18-20, 2014. pp. 41–48 (2014)
13. Normantas, K., Vasilecas, O.: Extracting term units and fact units from existing databases using the Knowledge Discovery Metamodel. Journal of Information Science 40(4), 413–425 (2014)
14. Object Management Group: Structured Metrics Metamodel (1 2012)
15. Oracle: SQL Developer. `www.oracle.com/technetwork/developer-tools`
16. Pérez-Castillo, R., de Guzmán, I.G.R., Caivano, D., Piattini, M.: Database schema elicitation to modernize relational databases. In: 14th International Conference on Enterprise Information Systems (ICEIS'12). pp. 126–132 (2012)
17. Pérez-Castillo, R., de Guzmán, I.G.R., Piattini, M.: Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems. Computer Standards & Interfaces 33(6), 519–532 (2011)
18. Sadovykh, A., Vigier, L., Hoffmann, A., Grossmann, J., Ritter, T., Gomez, E., Estekhin, O.: Architecture driven modernization in practice–study results. In: 14th IEEE International Conference on Engineering of Complex Computer Systems. pp. 50–57. IEEE (2009)
19. Wikimedia: Wikimedia dump index. `dumps.wikimedia.org/backup-index.html`