DBScribeHibernate: Documenting Database Usages and Schema Constraints in Database-Centric Applications using Hibernate

Bin Nie

1 ABSTRACT

Database-centric applications (DCAs) usually rely on database operations over a large number, even in thousands, of tables and attributes. So the development and maintenance of larger DCAs need the collaboration between developers and database administrators. However, due to the inherent gap in terms of skills and activities performed by the two roles and lack of detailed always-up-to-date documentations, the collaboration is challenging.

This project, **DBScribeHibernate**, aims at automatically generating always-up-to-date natural language descriptions of database operations and schema constraints in source code methods for Hibernate projects.

2 DBSCRIBEHIBERNATE DESIGN

2.1 Database-Related Method Types

The database-related methods in the target Hibernate project are grouped into three categories:

(1) SQL Operating Method
POJO class constructors and get/set methods

(2) Local SQL Method
Methods that call Hibernate Session built-in methods, such as Save(), Update(), Delete(), CreateQuery(), CreateCriteria(), etc.

(3) Delegated SQL Method
Methods that has at least one SQL operating method or local SQL method in its call-chain
Note that, the description of this method type is the *union* of its callees' method descriptions

Note that, method type (1) and type (2) are also called "*SQL Method*", and method type (3) is then called "*Higher-level SQL Method*".

2.2 Workflow

Figure1 shows the workflow for DBScribeHibernate.

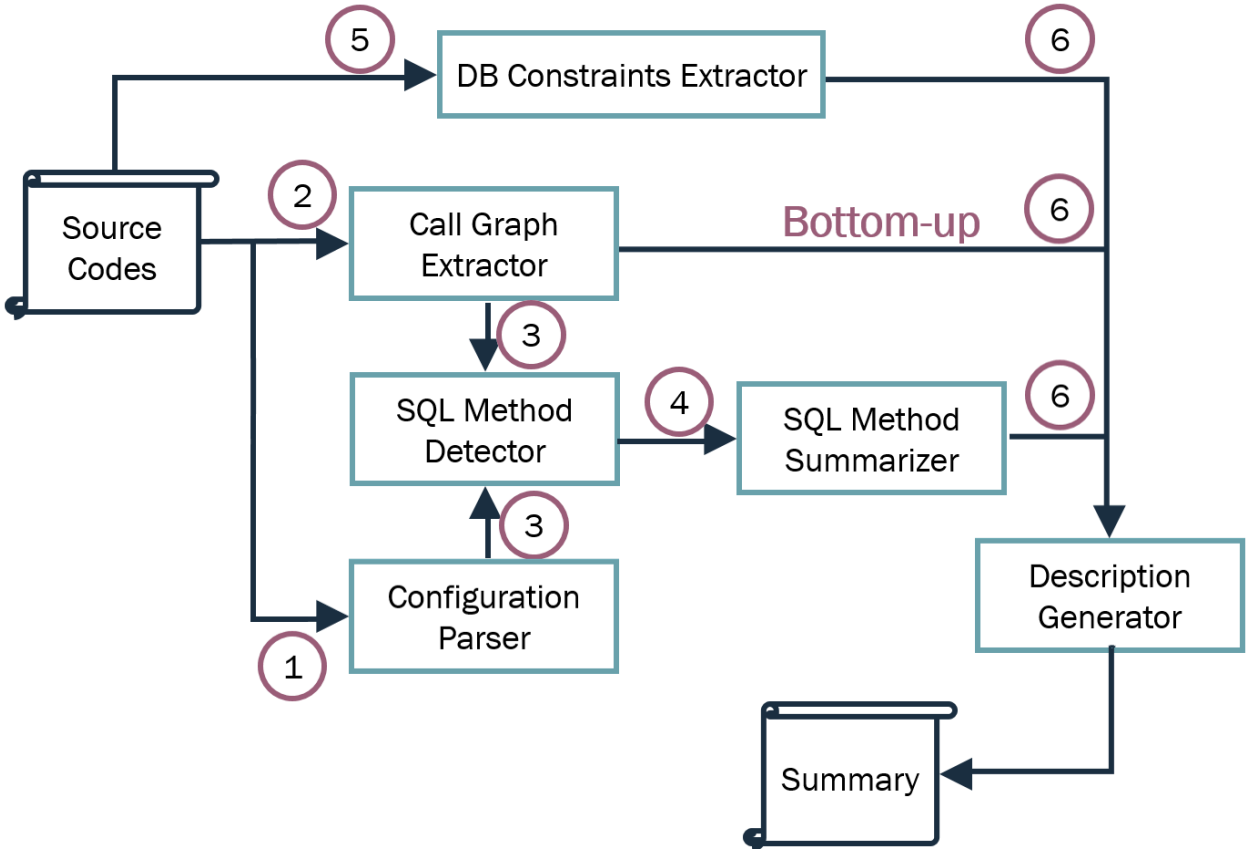

Figure 1 DBScribeHibernate Workflow

The input for DBScribeHibernate is the source code of the target Hibernate project. The output is a HTML-format report for database method's description and database constraints. As shown in figure1, there are mainly 6 steps:

(1) First of all, *Configuration Parser* analyzes the configuration files or annotations defined in the Java POJO classes to map POJO classes and their properties to corresponding database table and attributes.

(2) Secondly, *Call Graph Extractor* extracts caller list and callee list from the source code, and build call graph for the target project. It also topologically sorts the methods to prepare for later bottom-up method description propagation in step (6).

(3) Combining the previous information, *SQL Method Detector* finds *SQL Operating Methods* and *Local SQL Methods*.

(4) Then, *SQL Method Summarizer* can generate method description (such as which database table and attributes it interact with, what kind of operation) for each *SQL Method*.

(5) Next, *DB Constraints Extractor* retrieves database constraints from XML mapping files or from annotations defined in Java POJO classes.
(6) In the final step, *Description Generator* combines all the previously collected information, and traverses the call graph from bottom level to top level. For *SQL Operating Methods* and *Local SQL Methods*, it adds database constraints to their method descriptions. And for *Delegated SQL Methods*, it unions all its callee's method descriptions and database constraints.

3 MANIFEST

(1) DBScribeHibernate.rar → Source code
(2) DBScribeReports → Four HTML-format reports
(3) DBScribeHibernate_Documentation.chm → A documented class library for DBScribeHibernate, which contains the summaries for every class, field and method defined in the project.