

4M-4

Java プログラム内における SQL 文の書法と解析方法の提案*

岩城 賢治

小林 要†

金沢工業大学§

1 はじめに

近年、ネットワークを用いたデータベースアプリケーション(WebDB アプリケーション)が増えてきている。WebDB アプリケーションは、DB を操作する以外の処理も多く含まれ、複雑になりがちである。本研究では、Java 言語と SQL を用いた WebDB アプリケーションの静的解析について考察し、解析容易な SQL 文の書法について考察する。

WebDB アプリケーションが、どのような要求仕様の下に作成されたのか等の情報を得たい場合、手掛りになると考えられるのは SQL 文である。SQL 文には表名、属性名などが記述されているので、アプリケーションドメインの情報が多く含まれているからである。

2 Java と SQL 文の関係

WebDB アプリケーションは、Java プログラムと SQL 文が混在したソースプログラムとなっている。SQL 文を Java プログラムから DB に対して発行する方法として JDBC[1]と SQLJ[1]がある。Java・JDBC または Java・SQLJ と、複数の文法が混在することが考えられる。複数の文法が混在している文の種類には、コメント型・名前空間型・部分文字列型がある[2]。

Java・JDBC の組み合わせは、部分文字列型に分類される。部分文字列型に分類される理由として、Java 言語内に別の言語である、SQL 文を生成する文が入っているためである。Java・JDBC の組み合わせは、SQL 文が発行されている場所を特定することができる。

Java・SQLJ の組み合わせは部分文字列型・名前空間型に分類され、Java 文法を拡張させて解析をおこなう。部分文字列型に分類される理由として、Java 言語内に別の言語である、SQL 文を生成する文が入っているためである。Java・SQLJ の組み合わせが名前空間型にも

分類される理由として、Java 言語と SQLJ 文の境界が容易に識別できるためである。SQLJ 文は、トークン #sql で始まり、セミコロンで終了する。Java・SQLJ の組み合わせも、SQL 文が発行されている場所を特定することができる。

3 変数の還元

JDBC と SQLJ 内の SQL 文には、SQL 文を構成する文字列定数の他に、Java プログラム内の文字列変数が含まれている場合がある。文字列変数が含まれているため、正規の SQL 文のみを解析する文法の範囲外となる。SQL 文を解析するためには、文字列変数が還元できるかできないかが重要である。文字列変数の還元には、データフロー解析における“生きている変数”の解析アルゴリズム[3]などを使用する。

文字列変数の還元処理をおこなっても、文字列変数が文字列定数に還元できないならば、一つの SQL トークンを参照している文字列変数であると仮定して、解析をおこなうことを考える。解析を行うためには、文字列変数も含めて解析できる拡張 SQL 文法が必要となる。SQL92[4]の文法の<host identifier>を拡張し文字列変数が含まれる文法を例 1 で表す。文字列変数の前に”\$”をつける。

例 1: 拡張した SQL 文法の例

```
<host identifier> ::= <Ada host identifier>
| <C host identifier> | <COBOL host identifier>
| <Fortran host identifier> | <MUMPS host identifier>
| <Pascal host identifier> | <PL/I host identifier> | <New host identifier>
<New host identifier> ::= “$” <identifier>
```

4 部分文字列型の解析

解析をおこなう際に、ソフトウェア加工システム RPS[2]を使用する。RPS は、3 種類の機能がある。1 つ目の機能は、入力するテキストを入力テキストにあわせた文法で構文木を生成する機能。二つ目の機能は、

*A Method of Extracting SQL statement from Java programs and A proposal on SQL programming style.

†Kenji Iwaki, Kaname Kobayashi

§Kanazawa Institute of Technology

7-1 Ohgigaoka, Nonoichi, Ishikawa 921-8501, Japan

生成された構文木を加エールに沿って構文木を変換する機能。3 つ目の機能は構文木を出力文法にあわせテキストにする機能である。

部分文字列型の解析の流れを図 1 で説明する。

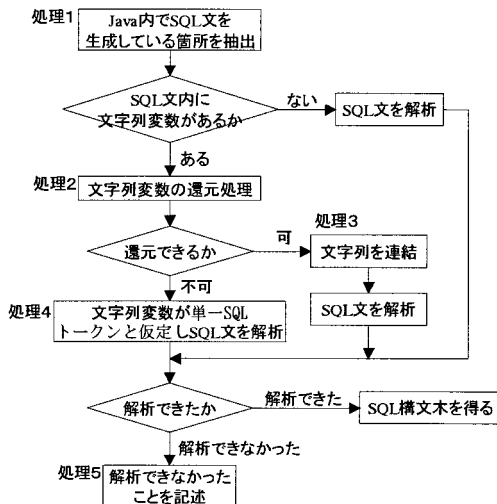


図 1: 部分文字列型での処理の流れ

部分文字列型の解析をおこなう前提として、Java プログラム内に SQL 文が含まれているテキストを、Java 文法をもとに構文解析し、構文木を生成しておく。

生成された構文木から、SQL 文が生成されている箇所、例えば JDBC では `executeQuery` メソッドや `execute` メソッドが記述されている箇所を抽出する(処理 1)。

文字列変数を還元するために、“生きている変数”の解析アルゴリズム[3]など用いて、できる限り文字列変数が参照する値を追跡する。(処理 2)

文字列変数が還元されるならば、SQL 文を構成する文字列定数と還元された文字列定数を連結して、1 つの SQL 文とする。(処理 3)

文字列変数が還元できない場合は、文字列変数が単一の SQL トークンを参照しているものと仮定し、解析を進めていく。(処理 4)

部分文字列型での SQL 文の種類

- ・すべてが文字列定数
- ・文字列定数と還元された文字列定数の連結
- ・文字列変数と文字列定数の連結

各 SQL 文を、文字列変数も含めて解析できる拡張 SQL 文法で解析をおこない、SQL 構文木を作成する。解析時にエラーが生じたときは、解析時に失敗したことを構文木に記録する。(処理 5)

5 SQL 文の書法

部分文字列型解析で、還元できない文字列変数は一つの SQL トークンであると仮定した。実行時に文字列変数が参照する文字列に、複数の SQL トークンが含まれている場合は、解析結果と実行時の動きが異なってくる。文字列変数 `UserTB` が単一 SQL トークンであると仮定したのが図 2 である。

```
executeQuery(SELECT 文(属性:"*"),FROM 文(表名:変数(変数名:"UserTB")))
```

図 2: 単一 SQL トークンと仮定した構文木

仮に文字列変数 `UserTB` が参照する文字列を、複数の SQL トークンを含んだ”`UserTable WHERE ID = 10`”であるとするなら、仮定の上で解析した結果は、`WHERE` 句の情報が抜け落ちてしまう。SQL 文内で文字列変数が含まれる場合は、文字列変数が参照する値を単一の SQL トークンに区切ることを SQL 文の書法として提案する。

6 おわりに

WebDB アプリケーションを解析する上で、重要と考えられる SQL 文を解析する方法を述べた。

文字列変数が SQL 文内で単一の SQL トークンを参照している変数であると仮定することで、変数を含む SQL 文においても解析することが可能となった。

SQL 文内における文字列変数の使用を、単一の SQL トークンを参照するものとする SQL 文の書法を提案した。

参考文献

- [1]二階堂 隆, 他: Oracle8i 詳細技術解説講座, 1999 年
- [2]小林 要: 文法や加工手順を入力データとするテキスト加工システム RPS の開発, 第 63 回情報処理学会全国大会 6Q-02, 2001 年
- [3]A.V.エイホ・J.D.ウルマン: コンパイラ, 1986 年
- [4]<http://aji.edd.osaka-sandai.ac.jp/doc/HOWTO/translations/fr/html/PostgreSQL-HOWTO-42.html>, 2001 年 7 月 27 日現在