C言語埋め込みSQL用のデバッガの開発

1H-8

柏原 幸一、竹元 賢也、中山 敬、川上 英
沖電気工業株式会社

1. はじめに

関係データベース操作言語SQL[1]は、1988年にJIS化され、実際に商用プログラム開発に用いられるようになりつつある。このSQL文のデバッグにはSQLインタプリタが使用されていた。しかし、SQLインタプリタは、DBMSへの対話インタフェースを目的に作られているものであり、埋込みSQLのデバッグという点では多くの問題がある。この問題を解決するために、筆者らはOKITAC8300上で動作するリレーショナルDBMS"REAM"[2][3]をアクセスするプログラムを効率的にデバッグするSQLデバッガを開発した。

本稿では、このSQLデバッガの設計方針と機能を報告する。

2. 埋込みSQLによるプログラム開発

SQLは、非手続き的な言語であり、プログラムの生産性が高いとされている。しかし、SQLを使って正しい問い合わせを書くのは習熟が必要である。このため、プログラム開発にあたっては、SQLインタプリタでSQL文のセマンティックを確認して、埋込みSQLプログラムを書くという手続きを取っていた。SQLインタプリタは埋込みSQLのデバッグを目的にして作られているため、埋込みSQLのデバッグに使用するには、以下に挙げるの問題点がある。

(1)全てのSQL文をサポートしていない。
　表1を参照。
(2)SELECT文において埋込みSQL文とセマンティックが異なる。
　インタプリタでは、検索レコードが複数存在してもエラーとならないが埋込みSQLでは、エラーとなる。
(3)カーソル操作ができない。
　FETCH文、UPDATE文:位置づけ、DELETE文:位置づけ使った処理の実行ができない。
(4)ホスト変数が使えない。

さらに、埋込みSQLでプログラム開発を行うときには、実際にコンパイルして実行イメージを作成しないと、呼び出しシーケンスの誤りとかINTO句で指定されているホスト変数の型の誤りなどのバグはわからないのが現状である。

また、SQLインタプリタの構文から埋込みSQLの構文に変更する時にバグが混入する可能性がある。このように、現状ではSQLでのプログラムの生産性は上がらない。

そこで、筆者らは埋込みSQL文のセマンティックだけではなく、シンタックスやシーケンスも効率的に検証できるSQLデバッガを作成した。

表1 SQLｲﾝﾀﾌﾟﾘﾀにおけるSQL文の有無

埋込みSQL	SQL ｲﾝﾀﾌﾟﾘﾀ
CLOSE文	×
COMMIT文	○
DELETE文:位置づけ	×
DELETE文:探索	○
FETCH文	×
INSERT文	○
OPEN文	×
ROLLBACK文	○
SELECT文	△
UPDATE文:位置づけ	×
UPDATE文:探索	○
埋込みSQL宣言	×
カーソル宣言	×
埋込み例外宣言	×

3. SQLデバッガの概要

SQLデバッガは、以下のような特徴を持っている。

- ソースプログラムをそのまま実行できる。
- コンパイルすることなく対話的に実行できる。
- すべてのSQL文をサポートしている。
- セマンティックだけではなく、シンタックスやシーケンスの検証ができる。
- ホスト変数に対して値の設定・参照ができる。
- 検索条件に変数が使える。

4. 構成

SQLデバッガは、図1に示されるようにコマンド処理部、SQL文解析部、SQL文実行部、SQLスタックの4つから構成される。

コマンド処理部:
　利用者の要求に従ってSQL文の読み込み、実行の制御、SQL文の表示、ホスト変数の値の設定、参照などを行う。

SQL文解析部:
　コマンド処理部の要求に従ってSQLスタックからSQL文を取り出し、シンタックスのチェッ

An Implementation of an interactive debugger for C-embedded SQL language
by K.KASHIWABARA, K.TAKEMOTO, T.NAKAYAMA and S.KAWAKAMI
Oki Electric Industry Co., Ltd.

クを行って中間語を作成しＳＱＬスタックに格納する。

ＳＱＬ文実行部:

コマンド処理部の要求に従って指定されたＳＱＬ文かまたはその中間語をＳＱＬスタックから取り出して実行し、結果をコマンド処理部に返却する。

ＳＱＬスタック

ＳＱＬスタックは、ソースプログラムから抽出したＳＱＬ文を格納する。ＳＱＬスタックには、ＳＱＬ文と中間語の２種類で格納される。

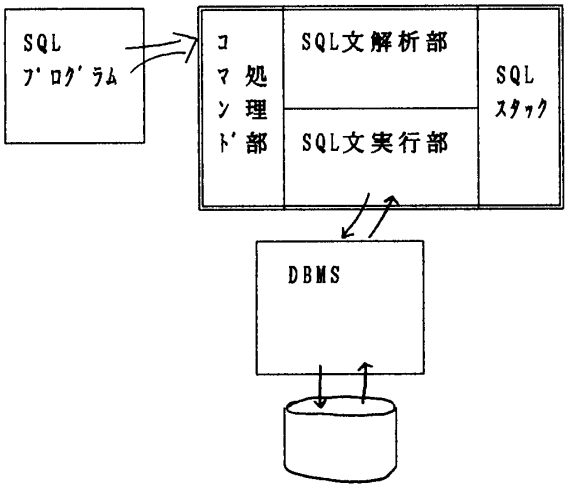

図１　ＳＱＬデバッガの構成

5. ＳＱＬデバッガの機能

ＳＱＬデバッガの機能を表２に示す。ＳＱＬデバッガは、埋込みＳＱＬソースファイルの中からＳＱＬ文のみを抽出し、シリアル番号を付けながらＳＱＬスタックに格納する。利用者は、図２で示されるようにそのシリアル番号を指定してＳＱＬ文の実行し、デバッグする。

表２　ＳＱＬデバッガコマンド一覧

ｺﾏﾝﾄﾞ	説明
exec	SQL文の実行を行う
quit	ﾃﾞﾊﾞｯｶﾞを終了する
read	ｿｰｽﾌｧｲﾙを読み込む
list	SQL文の表示
print	ﾎｽﾄ変数の表示
set	ﾎｽﾄ変数への値設定
his	ｺﾏﾝﾄﾞ発行履歴
man	操作説明

6. おわりに

本稿では、埋込みＳＱＬで記述したプログラムを、対話形式でデバッグできるＳＱＬデバッガに関して、その設計方針、システムの構成に関する報告を行った。本デバッガを用いることにより、親言語に埋め込まれたＳＱＬ文を、実際にコンパイルする前にデバッグすることが可能になり効率良い開発ができるようになった。今後は、ＳＱＬ言語の部分だけではなく親言語の部分もソースコードレベルで実行可能にすることでより使いやすくしていく予定である。

参考文献

[1] 工業技術標準調査会,"データベース言語ＳＱＬ",JIS X 3005,1989

[2] Kawakami S.,Nakayama T.,Kashiwabara K."REAM:An SQL Based and Extensible Relational Database Management System",Proc.DASFAA pp.166-170,1989

[3] 林,疋田,野原,三上,"OKITAC8300アーキテクチャーアプリケーションアーキテクチャ",沖電気研究開発 vol.55,No.4,pp12-14,1988

```
SDB>list
 ------------------------------ line 1
EXEC SQL BEGIN DECLARE SECTION;
char X[10];
int Y;
EXEC SQL END DECLARE SECTION;
 ------------------------------ line 2
EXEC SQL DECLARE C1 CURSOR FOR
      SELECT NAME
      FROM EMP
      WHERE EMP_NO = :Y;
 ------------------------------ line 3
EXEC SQL OPEN C1;
 ------------------------------ line 4
EXEC SQL FETCH C1 INTO: X;
SDB> EXEC 2
 ------------------------------ line 2
EXEC SQL DECLARE C1 CURSOR FOR
      SELECT NAME
      FROM EMP
      WHERE EMP_NO = :Y;
SDB> SET Y = 110
SDB> PRINT Y
Y        INT      :110
SDB> EXEC 3,4
 ------------------------------ line 3
EXEC SQL OPEN C1;
 ------------------------------ line 4
EXEC SQL FETCH C1 INTO: X;
X         CHAR(10)     :YAMADA
SDB>
```

図２ ＳＱＬデバッガの実行例