実践報告

SQLJプログラミング

ポリテクセンター京都
（京都職業能力開発促進センター）
秋田　正秀

1．はじめに

Oracleデータベースは，Oracle8iバージョンからJServerという名前でJava仮想マシンを搭載しています。これは，JavaのプログラムをOracle8iのサーバ側で実行できることを意味しています。加えて，Oracleデータベースの新たなプログラム開発手法にSQLJというものが導入されています。SQLJは，現在，ANSIやISOに標準規格として提出されています。SQLJは，Javaをホスト言語にした埋め込みSQLの規格です。

本内容では，Oracle8iを対象にしたSQLJプログラミングについて紹介します。Oracle8i JVMとSQLJについて概要を説明した後，SQLJのプログラム開発方法について紹介します。

2．Oracle8i JVM

Oracle8iからJava仮想マシンが搭載されました。Oracle8i JVM（Oracle8i Java Virtual Machine）は通常のJVMと比較してサーバ用に書き直されています。このOracle8i JVMはJava2対応になっています。Oracle社は，Sunが提供しているJava仮想マシンではスケーラビリティに問題があると指摘し，これをOracle社なりに改良しています。改良点は，以下のとおりです。

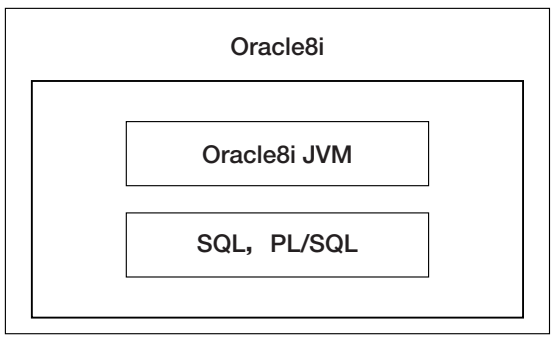

図1　Oracle8i JVM

・サーバ側によるマルチスレッド対応
・ガベージコレクタの最適化
・記憶領域管理の最適化
・メモリフットプリントの最小化
・プログラム実行の高速化

Oracle8i JVMはマルチユーザ対応に改良されており，Javaで作るマルチスレッドプログラムにまつわる問題を解決しています。通常のJVMでは，マルチユーザ対応になっていないのでマルチユーザに対応するためマルチスレッドのコードを作成します。Oracleでは，マルチスレッドのプログラムには２つの問題点があると指摘しています。１つは正しいマルチスレッドのプログラムの作成するのが難しいこと，もう１つはマルチスレッドのプログラムは共通のガベージコレクタを使うようになっており，複数のユーザでプログラムを使い始めるとガベージコレクタの処理が効率的ではなくなることです。Oracle8i JVMは，マルチユーザ対応になっているので作成するJavaプログラムはマルチスレッドを使わなくてよく，逆にシングルスレッドで作成するように推奨されています。

記憶領域管理では，記憶領域をコールメモリの新

領域と旧領域，セッションメモリの3つの領域に分け管理され，それぞれ違ったガベージコレクションのアルゴリズムが使われています。3つの記憶領域は，簡単には，静的変数の領域と使用頻度の高い変数の領域，使用頻度の低い変数の領域です。使用頻度の低い領域は頻繁にガベージコレクションの処理が行われ，使用頻度の高い変数の領域はガベージコレクションの処理の頻度が少なくなります。静的変数の領域は，プログラムやコールの終了時にガベージコレクションの処理が行われるだけになります。

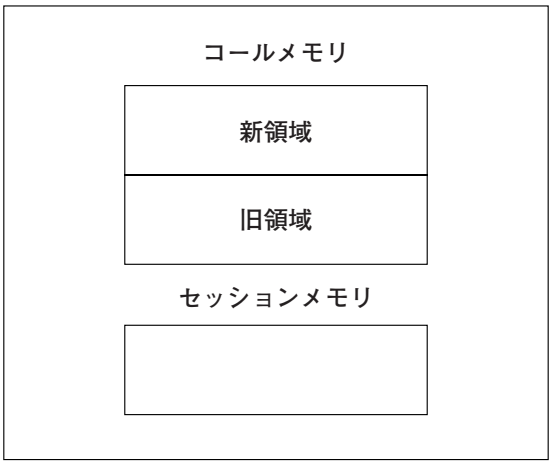

図2　ガベージ・コレクション

メモリフットプリントの最小化に関しては，Oracle8i JVMでは，UNIXをまねて同じセッション間でプログラムコードを共通の読取専用コードとして使うなどして，メモリフットプリントを最小限にとどめています。

プログラム実行の高速化では，実行コードを最適化されたネイティブコードにコンパイルして実行しています。通常，Javaのコードを実行するときにネイティブコードにコンパイルする技術としてJIT（Just-in-Time）コンパイラがありますが，最適化されたネイティブコードを生成するところが違います。最適化されたネイティブコードを生成するため，JITコンパイラと比較してコンパイルに時間がかかるようですが，何度も実行されるのであればこちらの方が速度的に有利になってきます。つまり，作成するプログラムがサーバサイドのものを想定し，1回きりではなく，何度も実行されると考えています。加えて，プログラムでよく使われるコアのクラスライブラリをネイティブコードで提供してプログラムの高速化をしています。

3．プログラムの種類

OracleデータベースアクセスのJavaプログラムは大きく3つに分類できます。1つ目は，Javaコマンドを実行してOSから起動するプログラムです。通常，アプリケーションと呼ばれています。2つ目は，ブラウザから起動するプログラムです。通常，アプレットと呼ばれています。3つ目は，データベースサーバ上で実行されるプログラムです。通常，ストアドプログラムやストアドプロシージャと呼ばれています。Oracle8iでは，これらすべての内容をJava言語1つでできることになりました。

4．データアクセス規格

プログラムからデータベースにアクセスするために規格があります。有名なものには，ODBC（Open DataBase Connectivity）があります。ODBCは，マイクロソフトが提唱した規格でWindowsには必ず入っています。一方，Java言語に対応したデータベースアクセスの規格にJDBCがあります。JDBCは，実際には，Javaのクラスになっており，JDBCドライバといわれます。JDBCドライバは，使われる形態によって4種類に分けられており以下のとおりです。

・JDBC-ODBCブリッジ
・ネイティブブリッジ
・ネットドライバ
・ダイレクトドライバ

Oracleでは，ネイティブブリッジとダイレクトドライバ，合わせてサーバドライバの3種類を提供しています。サーバドライバは，上記の分類からするとネイティブブリッジに分類されます。

Oracleが提供するネイティブブリッジには，従来からOracleデータベースにあるOCI（Oracle Call

Interface）というＣ言語で作られたAPIにマッピングするドライバと先ほど紹介したサーバドライバでKPRBというサーバ側に存在するＣ言語で作られたライブラリにマッピングする２つのドライバがあります。ダイレクトドライバは，すべてJavaで書かれたドライバでOracle8iを構成する１つのNet8まで記述されたドライバです。 そもそもクライアントからOracleデータベースにアクセスする場合，通信用のソフトウェアであるSQL＊NetやNet8が必要となりますが，このドライバを使うとNet8のインストール作業など必要なくなります。

５．SQLJ

ここでは，SQLJについて紹介します。SQLJは，Java言語をホスト言語にした埋め込みSQLの規格です。

５.１　概要

SQLJは，新しい埋め込みSQLの規格です。前で紹介したJDBCと関係が深く，JDBCをベースにして作られています。ただ，SQLJ自体は新しい規格ですが，この埋め込みSQL自体の発想はすでに存在しており，OracleデータベースではPro＊ＣやPro＊C++などがあります。SQLJが新しいこととしては，ホスト言語にJavaを採用したことと合わせて標準化したことです。SQLJ規格作成に参加した企業は，Oracle，IBM，Sybase，Tandem，Javasoftなどです。現在，規格をISOやANSIに提出している段階です。SQLJのホームページがあるので興味のある方はそちらを参照してください[1)]。

５.２　機能

SQLJには，大きく２つの機能があります。１つはプリコンパイラとしての機能，もう１つは実行時のAPIとしての機能です。プリコンパイラとしては埋め込みSQLの記述をJavaのソースコードに変換する機能です。プリコンパイラにオプションがあり，オプションによってはプリコンパイル時にデータベースに接続しセマンティクス（意味）のチェックをすることもできます。プリコンパイラから生成された埋め込みSQLの内容は，SQLJ RuntimeのAPIを使ったプログラムに変換します。

もう１つの機能は，SQLJ RuntimeとしてAPIの機能です。これは，プログラム実行時にプログラムから利用されるライブラリです。SQLJのソースをプリコンパイルするとSQLJ Runtime APIを使ったプログラムに変換され，プログラムを実行すると実行時にSQLJ Runtimeが結果として使われることになります。

５.３　SQLJの位置づけ

SQLJは，JDBCをベースにして作られています。SQLJを使って作られたアプリケーションは，実行時SQLJのRuntimeを呼び出して実行し，さらにそのSQLJのRuntimeがJDBC APIを呼び出してデータベースにアクセスします。

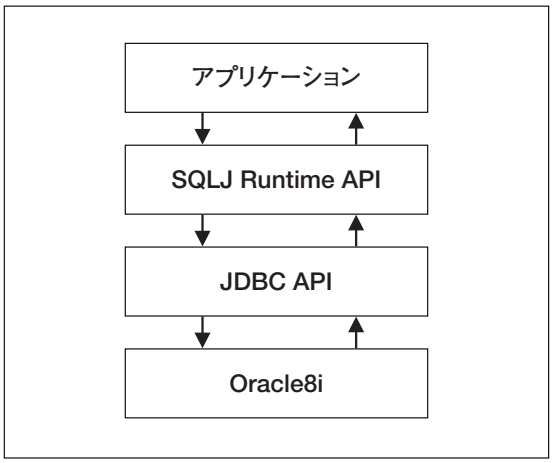

図３　SQLJ Runtime の位置づけ

６．SQLJプログラミング

ここでは，SQLJのプログラム作成方法をアプリケーション作成を通して紹介します。SQLJ文の要素，プリコンパイラの使い方，実行などを説明します。

６.１　SQLJプログラミングの基礎

ここでは，SQLJプログラム作成の基礎を紹介します。SQLJでは，静的な埋め込みSQLだけを扱います。動的な埋め込みSQLを扱う場合は，PL/SQLのパッケージを使うか，JDBCを使うことになりま

す。Oracle社の紹介でも，SQLJとJDBCは補完的に使うことを説明しています。ですから，SQLJによってJDBCが必要でなくなるわけではありません。開発する内容に合わせて適したものを選ぶことになります。

この節の残る部分でSQLJのソースを記述するSQLJ文について説明します。

⑴ 接続コンテキスト

SQLJのプログラムを実行するとき，どのデータベースにアクセスするかという情報が必要になります。その接続情報を管理しているものが接続コンテキストです。

⑵ イテレータ

イテレータはデータベースからデータを取り出すときに使います。基本的にはデータベースのカーソルと同じような働きをします。

⑶ SQLJ文の種類

SQLJ文は必ず先頭に#SQLがつきます。そして，SQLJ文には2種類のものがあります。1つは宣言文で，もう1つが実行可能文です。宣言文には，接続コンテキストとイテレータの宣言文があります。実行可能文には，SQL文を記述します。

6.2 プログラム作成

ここでは，アプリケーションの作成を通してSQLJプログラムの作成手順を紹介します。サンプルを付録1に示します。

⑴ ソースの作成

はじめに，SQLJ文を含んだJavaのソースファイルを作成します。先ほど紹介した宣言文と実行可能文を使ってソースを作成します。ソースファイルの拡張子はSQLJです。また，Javaのコンパイルを考えると自動的に主ファイル名も決まってきます。

⑵ プリコンパイラ

SQLJのソースができたらプリコンパイラでプリコンパイルし，完全なJavaソースファイルを作成します。このときのオプションにユーザ名を使うと，セマンティクスのチェックも事前にすることができます。また，同時にJavaソースファイルのコンパイルをすることもできます。トランスレータのコマンドは，SQLJ.EXEです。一例ですが，サンプルプログラムファイルをプリコンパイルします。コマンドは，以下のように実行します。

```
SQLJ COMPILE=FALSE TestSQLJ.sqlj
```

実行するとORACLE_HOMEを問い合わせてくるので表示されているORACLE_HOMEのパスがあっていればそのままY（YES）を，違っていれば環境変数のORACLE_HOMEのパスを変更します。

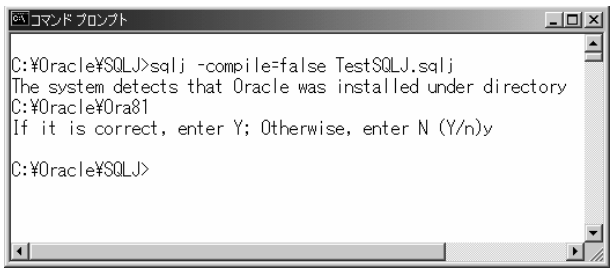

図4 プリコンパイル

プリコンパイルが成功すると2つのファイルが作成されます。作成されるファイルは以下の2つです。

・TestSQLJ.java
・TestQLJ_SJProfile0.ser

1つがJavaの完全なソースファイルで，もう1つはプロファイルです。Javaソースファイルは，SQLJソースファイルにあったSQLJの宣言文と実行可能文をJavaのソースコードに変換したものです。ですから，変換後には#sqlの記述はJavaのソースに変換され無くなっています。一方，プロファイルはデータベース固有の情報が入っているファイルです。例えば，データ型などです。ファイル自体は，バイナリでJavaのクラスファイルと同じようなものです。実際，プロファイルをクラスファイルとして

生成することもできます。

このあと，Javaソースファイルをコンパイルして Javaのクラスを生成しでき上がりとなります。

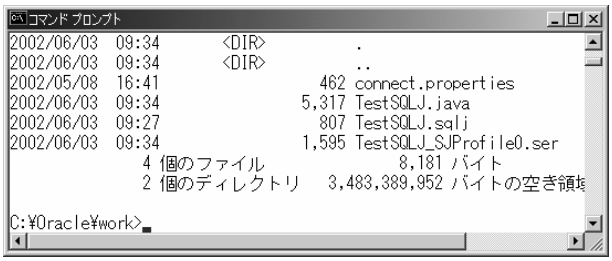

図5　生成されたファイル

6.3　プログラムの実行

このサンプルプログラムでは接続情報をCONNECT.PROPERTIESというファイルから取り込むように指示しています。

```
Oracle.connect (TestSQLJ.class,connect.properties);
```

このCONNECT.PROPERTIESファイルはテキスト形式でユーザ名やパスワード，接続するJDBCドライバの指定，接続文字列が記述されます。

使った内容を以下に示します。

```
sqlj.url=jdbc:oracle:oci8:@kyoto1
sqlj.user=scott
sqlj.password=tiger
```

sqlj.urlでは，JDBCドライバの種類でOCI8と接続文字列KYOTO1を設定しています。sqlj.userではユーザ名，sqlj.passwordではパスワードを指定しています。ここでは，OCIライブラリを使うJDBCドライバを指定しており，接続文字列にkyoto1を使っています。ログインユーザ名がscottでパスワードがtigerとなっております。

プログラムを実行します。

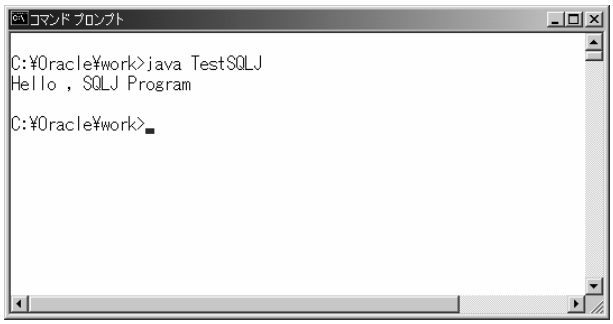

図6　プログラム実行

7．終わりに

SQLJは，新しくできた規格でこれからのものです。現在，SQLJのホームページを見るとSQLJが規格化されるためにクリアしなければならない問題がありそれをPDFファイルで公開しています。興味のある方は見てください。Oracle8iでは，これまでにもOracleデータベースのためのプログラム開発手法を提供してきました。しかし，クライアントからサーバサイドまで１つの言語で対応できるようになったのはJavaがはじめてです。本内容では，アプリケーションの作成を行いましたが，その他，アプレットやストアドプロシージャを作ることもできます。アプリケーションでは，Ｃ言語やＣ++言語をホスト言語としたものや，あるいは，Visual BasicでCOMを使ったものなどありますが，サーバサイドのプログラムをこれらでは作成できません。一方，サーバサイドには，PL/SQLというストアドプロシージャを作る言語がありますが，PL/SQLではアプリケーションを作成することはできません。これらをひっくるめてJava言語は全部できるようになりました。開発言語を統一できるということは，例えば，プロジェクトでシステムを開発する場合，非常に有効になります。プロジェクトでのすべての会話がJavaで通用することになるからです。

しかし，一方でJavaを使っているデメリットもあります。例えば，PL/SQLで作ったストアドプロシージャで処理をさせるのと同じ内容をJavaで書いたストアドプロシージャで処理させるのではPL/SQLの方に軍配が上がります。

現在は，部分的にはこのようなデメリットもあり

ますが，コンピュータの高速化や実行環境の改善にも目覚しい進歩があります。今後，システム開発の有効な手段としてどんどん確立されると思います。

＜参考文献＞

1）URL：www.sqlj.org.
2）Oracle：Oracle JServer概要，Oracle資料.
3）Oracle：Javaによるストアド・プロシージャ作成，Oracleテクニカル・ホワイトペーパ，1998年11月.
4）Oracle：Javaストアド・プロシージャの概要及び開発方法，Oracle資料.
5）Oracle：JDBC/SQLJの概要及び開発技法，Oracle資料.
6）Oracle：SQLJ：Trick,Tips and Gems，Oracleテクニカル・ホワイトペーパ.
7）Oracle：AN OVERVIEW OF SQLJ:EMBEDDED SQL IN JAVA，Oracle資料.
8）Oracle：Embedded SQL in Java，Oracleテクニカル・ホワイトペーパ，1998年5月.
9）Oracle：JavaからOracleへのアクセス，Oracle資料.
10）Oracle：エンタープライズ・インターネット・プラットホームの作成 Oracle8iへのJavaの統合，Oracleテクニカルホワイトペーパ，1999年4月.
11）Oracle：Oracle8iJVMの仕組み，Oracle資料，1999年7月.
12）SE編集部訳：ORACLE8i SQLJプログラミング，翔泳社，2000年6月.


```
import java.sql.SQLException ;
import oracle.sqlj.runtime.Oracle;

#sql iterator TestIter (String FD1);

class TestSQLJ
{
  public static void main (String args[])
  {
    try{
      Oracle.connect(TestSQLJ.class, "connect.properties");

      TestSQLJ ts = new TestSQLJ();
      ts.runSQLJ();
    }
    catch (SQLException e){
      System.err.println("エラー:" + e);
    }
    finally{
      try { Oracle.close(); } catch (SQLException e) { }
    }
  }

  void runSQLJ() throws SQLException
  {
    #sql { DELETE FROM FORSQLJ};
    #sql { INSERT INTO FORSQLJ(FD1) VALUES ('Hello , SQLJ Program')};

    TestIter iter;
    #sql iter = { SELECT FD1 FROM FORSQLJ };

    while (iter.next()) {
      System.out.println(iter.FD1());
    }
  }
}
```

付録1　サンプルプログラム