Micro Focus Visual COBOL 2.2J x64/x86 Linux

PostgreSQL データアクセス

動作検証 検証結果報告書

2013 年 12 月 26 日

マイクロフォーカス株式会社

1. 検証概要、目的及びテスト方法

1.1 検証概要

PostgreSQL はカルフォニア大学バークレー校で開発された POSTGRES, Version 4.2 をベースにしたオープンソースのリレーショナルデータベース管理システムです。PostgreSQL はオープンソースでありながら、商用リレーショナルデータベース管理システム製品に劣らない機能を装備しています。そのため、学習用途に限らず企業システムにおいても利用されるケースも往々にしてございます。更に近年の企業システムへのオープンソースソフトウェア活用の流れに相まって一層注目を集めるリレーショナルデータベース管理システムです。

Visual COBOL に付属する OpenESQL プリプロセッサは、COBOL プログラムに記述された埋め込み SQL 文より ODBC ドライバ、JDBC ドライバ、ADO.NET データプロバイダを経由した様々なリレーショナルデータベースアクセスを提供します。本稿では、この OpenESQL を使って ODBC 並びに JDBC 経由で、埋め込み SQL 文を含む COBOL プログラムから PostgreSQL へアクセスできることを動作検証しました。

1.2 目的及びテスト方法

Micro Focus Visual COBOL は最新鋭の COBOL 言語開発・実行環境を提供します。COBOL 言語への埋め込み SQL 処理系を標準装備しており、ODBC ドライバ、JDBC ドライバ、ADO.NET データプロバイダを経由した様々なデータベースへのアクセスを可能とする OpenESQL プリプロセッサを搭載します。製品出荷時に弊社にて動作検証できているのは Oracle, SQL Server, DB2 のみですが、OpenESQL を使えば ODBC については ODBC 3.x 仕様に、JDBC であれば JDBC 4.0 仕様に準拠したデータソースに対して設計上問題なくアクセスすることができます。今回、本稿執筆時点における最新版 PostgreSQL 9.3.1 に対して COBOL プログラムより日本語を含むデータを正しく操作できることを検証しました。ODBC 経由のアプリケーションについては、Linux の Native コードにコンパイルされた動的ロードモジュールより処理を実行しています。JDBC 経由で接続するアプリケーションは、javabyte コードにコンパイルし、JVM クラスとして java コマンドから実行して動作を確認しています。

2. 検証環境

ソフトウェア

Red Hat Enterprise Linux 6 Update 4(VM のゲスト OS として稼働)
PostgreSQL Server 9.3.1
iODBC ドライバマネージャ 3.52.0812.0326

PostgreSQL ODBC ドライバ 9.02.0100
Simple-JNDI 11.4.1
PostgreSQL JDBC ドライバ 9.3-1100 JDBC 41
Micro Focus Visual COBOL 2.2J for x64/x86 Linux

ハードウェア

機種：	Dell OPTIPLEX7010
CPU：	Intel Core2 i7-3770 3.40GHz
Memory：	2.00 Gbyte memory(ゲスト OS に割り当てたサイズ)

3. テスト内容

COBOL プログラム中に CREATE TABLE 文を埋め込み SQL 文として記述し、テスト用のテーブル作成します。続いて、INSERT 文によるデータの充填、UPDATE 文によるデータの編集を行います。INSERT 文、UPDATE 文の後には COMMIT 文を入れそれぞれのトランザクションを確定させます。扱うデータには日本語を含めます。反映したデータは CURSOR – FETCH して取り出し、中身を確認します。最後に DROP TABLE 文を使って作成したテーブルを削除します。これにより、DDL 文、DML 文、DCL 文の正常動作並びに日本語データの正常なハンドリングを検証します。

4. 結果

4.1 インストール

> PostgreSQL Server
Red Hat 6.4 は 8.4 をバンドルしているため、下記リンク先より 9.3 用の RPM をダウンロードの上、yum リポジトリを更新し yum コマンドでインストールしました。

ダウンロード元(2013/12/4 リンク検証)：
http://yum.postgresql.org/repopackages.php

> iODBC ドライバマネージャ
以下のリンク先よりソースをダウンロードし、コンパイル・インストールしました。

ダウンロード元(2013/12/4 リンク検証)：

http://downloads.sourceforge.net/project/iodbc/iodbc/3.52.8/libiodbc-3.52.8.tar.gz

> PostgreSQL ODBC ドライバ
以下のリンク先よりソースをダウンロードし、コンパイル・インストールしました。

ダウンロード元(2013/12/4 リンク検証)：
http://www.postgresql.org/ftp/odbc/versions/src/

> Simple-JNDI
以下のリンク先よりダウンロードし、インストールしました。

ダウンロード元(2013/12/9 リンク検証)：
http://code.google.com/p/osjava/downloads/detail?name=simple-jndi-0.11.4.1.zip&can=2&q=

> PostgreSQL JDBC ドライバ
以下のリンク先よりダウンロードし、インストールしました。

ダウンロード元(2013/12/9 リンク検証)：
http://jdbc.postgresql.org/download.html

4.2 サンプルアプリケーションの作成

本検証では下記のような流れでデータベースアクセスするプログラムを作成して動作を確認しました。今回、ODBC 経由及び JDBC 経由による接続を検証していますが、CONNECT 文以外のロジックは共通のものを用いています。

① PostgreSQL データベースに接続
② CREATE TABLE 文にてテスト用のテーブルを作成
③ INSERT 文にて日本語を含まないデータを挿入
④ INSERT 文にて日本語を含むデータを挿入
⑤ COMMIT 文を発行してデータ挿入のトランザクションをコミット
⑥ UPDATE 文にて日本語を含むデータを編集
⑦ COMMIT 文を発行してデータの変更をコミット
⑧ DECLARE CURSOR 文にてテスト用のテーブルを参照するカーソルを定義
⑨ FETCH 文にてデータを取得

⑩　DROP TABLE 文にてテスト用に作成したテーブルを削除
⑪　PostgreSQL データベースとの接続を切断

実際に使用したプログラムは、Micro Focus のウェブ上に本報告書と供に公開しています。

4.3 サンプルアプリケーションの実行結果

Linux のネイティブアプリケーション並びに JVM クラスとして生成したサンプルアプリケーションを正常に実行できることを確認しました。詳細は付録の通りとなります。

5. テスト結果及び考察

埋め込み SQL 文を含む COBOL プログラムを Visual COBOL を使ってNative コードにコンパイルしたアプリケーションより PostgreSQL データベースに ODBC 経由で接続して DDL 文、DML 文、DCL 文を発行してデータベースを操作できることを検証しました。javabyte コードにコンパイルし、JVM クラスとした場合も 同様に JDBC 経由にて正しく操作できることを確認しました。また、日本語データについても両者ともに正常に扱えることも検証できました。

以上

付録 1. サンプルアプリケーションの実行 – ODBC 編

1) 構成ファイル odbc.ini に PostgreSQL データベースへアクセスするための DSN を設定

```
$ cat /etc/odbc.ini
[POSTGRES]
Driver = /usr/local/lib/psqlodbcw.so
Servername = localhost
Database = testdb
$
```

2) PostgreSQL サーバを起動

```
$ pg_ctl -D ~/9.3/data -l logfile start
サーバは起動中です。
$
```

3) iodbc が提供する接続ツール iodbctest を使って 1) で構成した DSN にて PostgreSQL データベースへ接続できることを確認

```
$ iodbctest "DSN=POSTGRES;UID=postgres;PWD=password"
iODBC Demonstration program
This program shows an interactive SQL processor
Driver Manager: 03.52.0812.0326
Driver: 09.02.0100 (psqlodbcw.so)

SQL>
```

4) 検証用に作成したプログラムをコンパイル

```
$ cob -u PSQLTEST0.cbl
$
```

5) プログラムを実行

```
$ cobrun PSQLTEST0.gnt
Create/insert/update/select/drop test

Create table
Insert first row
Insert second row containing Japanese characters
Commit the insertion
Update row
Commit the change
Fetch
***********************************
int_col     : +00001
varchar_col : Single byte chars
***********************************
int_col     : +00002
varchar_col : かきくけこ
***********************************
Drop table
Disconnect
Test completed without error
$
```

➔ 全て正常に処理できていることが確認できます。

付録 2. サンプルアプリケーションの実行 – JDBC 編

1) Java のバージョンを確認

```
$ java -version
java version "1.7.0_09-icedtea"
OpenJDK Runtime Environment (rhel-2.3.4.1.el6_3-x86_64)
OpenJDK 64-Bit Server VM (build 23.2-b09, mixed mode)
$
```

➔ PostgreSQL コミュニティサイトにより Java 1.7 を使う場合は、JDBC 41 版が推奨されている旨確認。
http://jdbc.postgresql.org/download.html(2013/12/9 リンク検証)

2) ダウンロードした JDBC Driver を CLASSPATH 環境変数に追加

```
$ export CLASSPATH=$CLASSPATH:~/jdbclib/postgresql-9.3-1100.jdbc41.jar
$
```

3) Simple-JNDI をセットアップ

① Simple-JNDI のライブラリを CLASSPATH に追加

```
$ export CLASSPATH=$CLASSPATH:/var/lib/pgsql/jdbclib/simple-jndi-0.11.1/simp
le-jndi-0.11.4.1.jar
$
```

② jndi.properties ファイルを作成

```
$ cat jndi.properties
java.naming.factory.initial=org.osjava.sj.SimpleContextFactory
org.osjava.sj.root=/var/lib/pgsql/jdbclib/jndidir
$
```

③ PostgreSQL を利用する Data Source 用の properties ファイルを作成

```
$ cat pg.properties
type=javax.sql.DataSource
driver=org.postgresql.Driver
url=jdbc:postgresql:testdb
user=postgres
password=password
$
```

④ properties ファイルの格納ディレクトリを CLASSPATH に追加

```
$ export CLASSPATH=$CLASSPATH:/var/lib/pgsql/jdbclib/jndidir
$
```

4) PostgreSQL サーバが起動中であることを確認

```
$ pg_ctl status
pg_ctl: サーバが動作中です(PID: 5882)
/usr/pgsql-9.3/bin/postgres "-D" "/var/lib/pgsql/9.3/data"
$
```

5) 検証用に作成したプログラムをコンパイル

```
$ cob -C"JVMGEN(MAIN)" PSQLTESTJ.cbl
$
```

6) プログラムを実行

```
$ java PSQLTESTJ
Create/insert/update/select/drop test

Create table
Insert first row
Insert second row containing Japanese characters
Commit the insertion
Update row
Commit the change
Fetch
***********************************
int_col     : +00001
varchar_col : Single byte chars
***********************************
int_col     : +00002
varchar_col : かきくけこ
***********************************
Drop table
Disconnect
Test completed without error
$
```

➔ 全て正常に処理できていることが確認できます。

以上