

Micro Focus Visual COBOL 2.2J

MySQL データアクセス

動作検証 検証結果報告書

2014 年 1 月 14 日

マイクロフォーカス株式会社

Copyright © 2014 Micro Focus. All Rights Reserved.

記載の会社名、製品名は、各社の商標または登録商標です。

1. 検証概要、目的及びテスト方法

1.1 検証概要

MySQL は 2 名のスウェーデン人技師並びに 1 名のフィンランド人技師によって設立されたスウェーデン企業 MySQL AB によって開発されたリレーショナルデータベースです。Sun Microsystems による MySQL AB の買収を経て現在では Oracle 社によって同製品は管理されています。1995 年に初期版がリリースされて以降、Yahoo、Facebook、Twitter など多くの企業で利用されており、PostgreSQL と並び世界で最も普及しているオープンソースデータベースの 1 つです。

Visual COBOL が備えるプリプロセッサ OpenESQL 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、ADO.NET データプロバイダ、JDBC ドライバを経由した様々なリレーショナルデータベースへのアクセスを可能とします。本稿では、この OpenESQL を使って ODBC、ADO.NET 並びに JDBC 経由で、埋め込み SQL 文を含む COBOL プログラムからこの MySQL 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。本製品は 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、ADO.NET データプロバイダ、JDBC ドライバ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、ADO.NET であれば CLR2 及び CLR 4 向けの ADO.NET データプロバイダに、JDBC であれば JDBC 4.0 仕様に準拠したデータソースであれば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 MySQL 5.6 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。ADO.NET 経由で接続するアプリケーションは、MSIL コードにコンパイルし .NET Managed アプリケーションとして動作を確認しています。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

➤ Database サーバ

ソフトウェア

- ・ Red Hat Enterprise Linux 6 Update 4(VM のゲスト OS として稼働)
- ・ MySQL Server 5.6.15-enterprise-commercial-advanced
- ・ unixODBC ドライバマネージャ 2.2.14
- ・ MySQL ODBC ドライバ Connector/ODBC 5.2.6
- ・ MySQL JDBC ドライバ Connector/J 5.1.28
- ・ Simple-JNDI 11.4.1
- ・ Micro Focus Visual COBOL 2.2J for x64/x86 Linux

ハードウェア

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

➤ Database クライアント(ADO.NET の検証に使用)

ソフトウェア

- ・ Windows 8 Enterprise Edition(VM のゲスト OS として稼働)
- ・ MySQL ADO.NET ドライバ Connector/Net 6.8.3
- ・ Microsoft Visual Studio Professional 2012 Version 11.0.60610.01 Update 3
- ・ Micro Focus Visual COBOL 2.2J for Windows

ハードウェア

機種 : Dell OPTIPLEX7010
CPU : Intel Core i7-3770 3.40GHz
Memory : 3.00 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

> MySQL Server

以下のリンク先より製品をダウンロードし、必要なモジュールをインストールしました。

ダウンロードした製品名：

MySQL Database 5.6.15 RPM for Oracle Linux / RHEL 6 x86 (64bit)

インストールに使用した RPM：

MySQL-server-advanced-5.6.15-1.el6.x86_64.rpm

MySQL-client-advanced-5.6.15-1.el6.x86_64.rpm

MySQL-shared-advanced-5.6.15-1.el6.x86_64.rpm

ダウンロード元(2014/01/10 リンク検証)：

<http://www.mysql.com/downloads/>

> unixODBC ドライバマネージャ

yum コマンドを使って以下のモジュールをダウンロード/インストールしました。

インストールモジュール：unixODBC.x86_64 2.2.14-12.el6_3

> MySQL ODBC 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。

ダウンロードファイル：mysql-connector-odbc-5.2.6-1.el6.x86_64.rpm

ダウンロード元(2014/01/10 リンク検証)：

<http://dev.mysql.com/downloads/connector/odbc/>

> ADO.NET 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。

ダウンロードファイル：mysql-connector-net-6.8.3.msi

ダウンロード元(2014/01/10 リンク検証) :

<http://dev.mysql.com/downloads/connector/net/6.8.html>

> 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(2014/01/10 リンク検証) :

<http://code.google.com/p/osjava/downloads/detail?name=simple-jndi-0.11.4.1.zip&can=2&q=>

> MySQL 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。

ダウンロードファイル : mysql-connector-java-5.1.28.tar.gz

ダウンロード元(2014/01/10 リンク検証) :

<http://dev.mysql.com/downloads/connector/j/>

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、ADO.NET、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 MySQL 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 MySQL 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共に公開しています。

これを実際に利用してお試しになる場合は接続部分を自身の環境に合ったかたちに置き換えてご利用ください。

4.3 サンプルアプリケーションの実行結果

接続部分を除いて同一のロジックを持つ COBOL プログラムから Linux のネイティブアプリケーション、.NET Managed コード並びに JVM クラスにコンパイルしました。それぞれ ODBC、ADO.NET、JDBC を経由して日本語を含むデータを正しく MySQL とやりとり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 を使ってコンパイルしたアプリケーションより MySQL データベースに接続して 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。今回は、Linux のマシンコード、MSIL コード、javabyte コードにコンパイルして検証しています。ここでは、コンパイルのターゲットに合わせてロジックを変えるのではなく、環境に依存する必要最低限の部分(今回の例では接続部分)を環境に合わせたのみで対処しており、Visual COBOL のポータビリティを傍証できました。また、日本語データについてもいずれについても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 MySQL データベースへアクセスするための DSN を設定

```
$ cat /etc/odbc.ini
[MYSQLO]
Driver          = /usr/lib64/libmyodbc5a.so
DataBase        = mysqlDemoU
SERVER          = localhost
SOCKET          = /var/lib/mysql/mysql.sock
$
```

- 2) MySQL サーバを起動

```
# service mysql start
Starting MySQL. SUCCESS!
#
```

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

```
$ isql MYSQLO yoshihiro password
+-----+
| Connected!          |
|                    |
| sql-statement      |
| help [tablename]   |
| quit               |
|                    |
+-----+
SQL>
```

- 4) 検証用に作成したプログラムをコンパイル

```
$ cob -u MYSQLTEST0.cbl
$
```

- 5) プログラムを実行

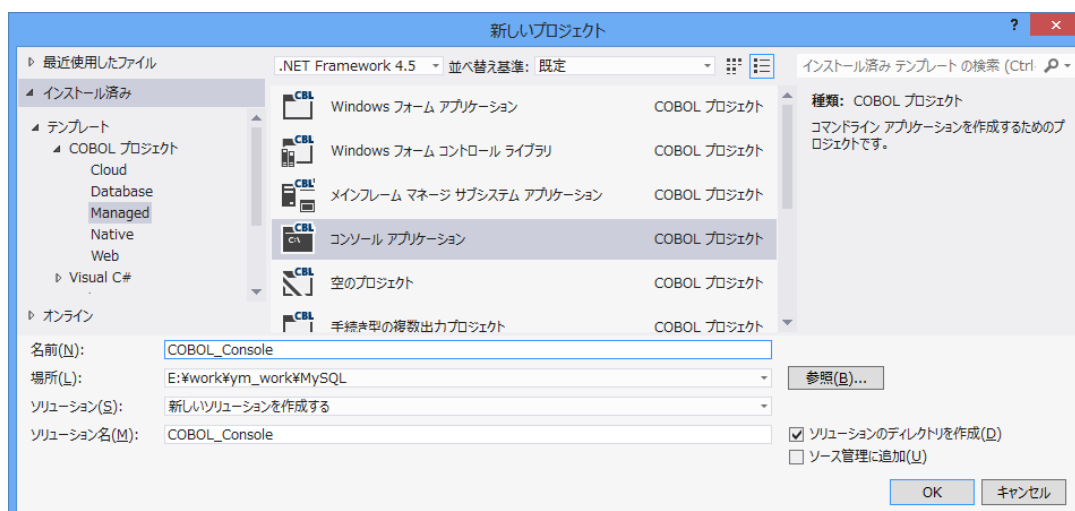
```
$ cobrun MYSQLTEST0.gnt
Create/insert/update/select/drop test

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。

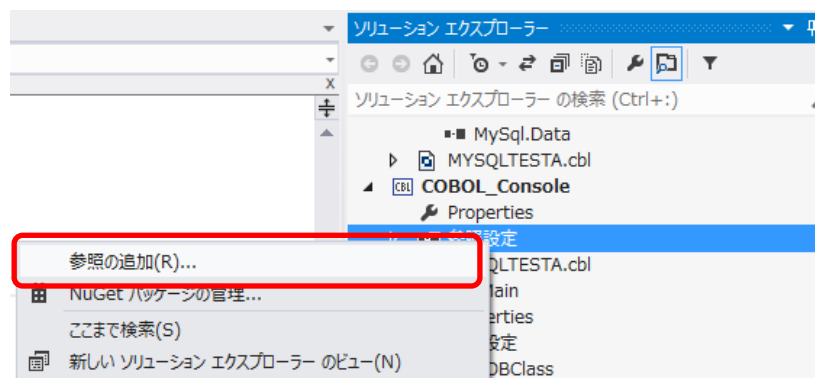
付録 2. サンプルアプリケーションの実行 – ADO.NET 編

1) COBOL の Managed コンソールアプリケーションプロジェクトを作成

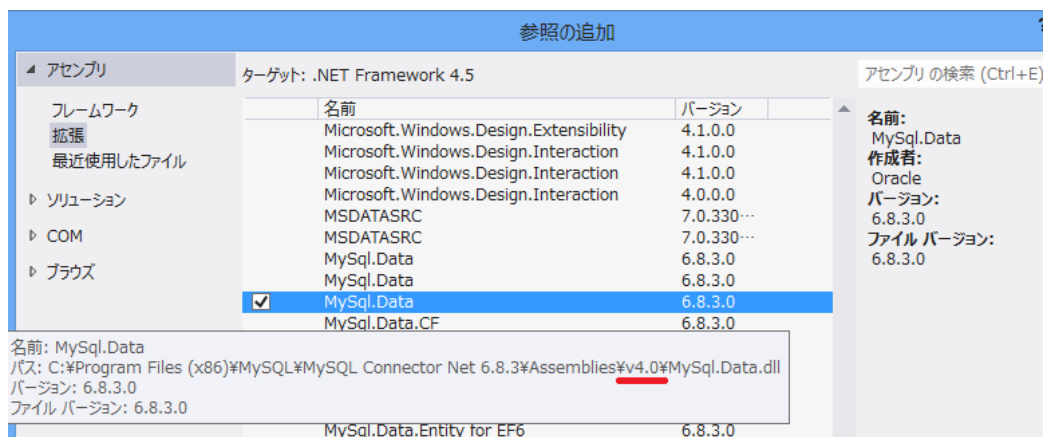


2) MySQL の ADO.NET ドライバを参照に追加

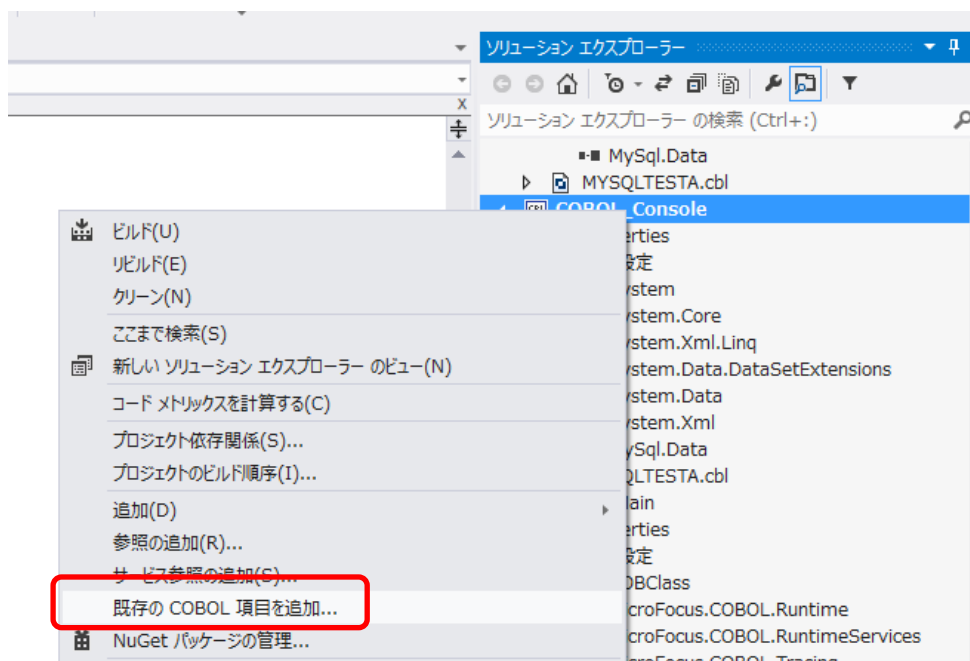
- ① [ソリューションエクスプローラ] にて [参照設定] を右クリックし、[参照の追加] を選択



- ② [アセンブリ] > [拡張] ページにて OpenESQL が保証する CLR 4.0 バージョンの MySQL ADO.NET ドライバを選択



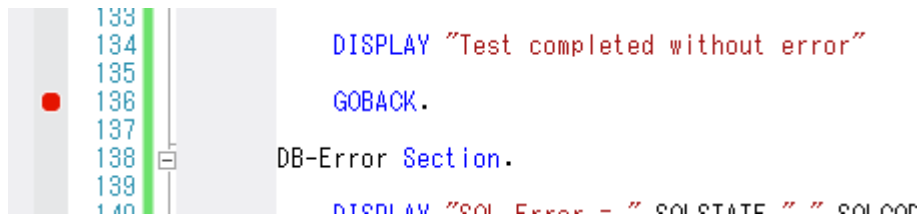
- 3) [ソリューションエクスプローラ]にてプロジェクトを右クリックから [既存の COBOL 項目を追加] を選択し、プロジェクトに検証用プログラム MYSQLTESTA.cbl をインポート



- 4) [ビルド] メニュー > [ソリューションのリビルド] を選択し、プログラムをコンパイル

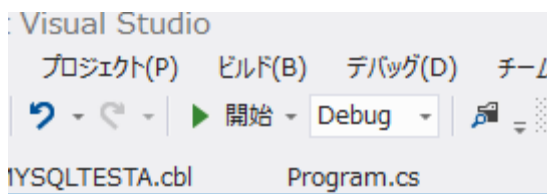


- 5) GOBACK 文にブレークポイントを指定

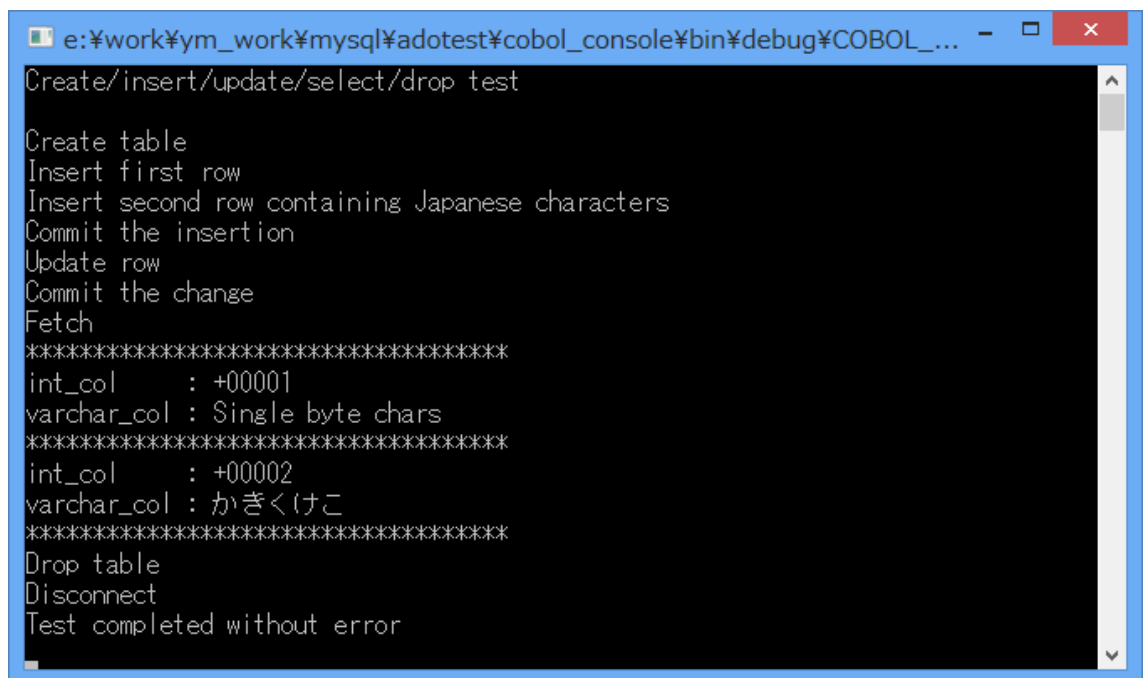


```
133  
134 DISPLAY "Test completed without error"  
135  
136 GOBACK.  
137  
138 DB-Error Section.  
139  
140 DISPLAY "SQL Error = " SQLSTATE " " SQLERRM
```

- 6) [開始] をクリックしてプログラムを起動



- 7) 正常にアプリケーションが処理されていることを確認



```
e:\work\ym_work\mysql\adotest\cobol_console\bin\debug\COBOL_...  
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
```

付録 3. 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)
$
```

➔ *Java 1.7 support requires Connector/J 5.1.21 and higher.*¹

にて組み合わせに矛盾がないことを確認しています。

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

```
$ export CLASSPATH=$CLASSPATH:`pwd`/mysql-connector-java-5.1.28-bin.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:`pwd`/simple-jndi-0.11.4.1.jar
$
```

② jndi.properties ファイルを作成

```
$ cat /home/yoshihiro/mylib/jndidir/jndi.properties
java.naming.factory.initial=org.osjava.sj.SimpleContextFactory
org.osjava.sj.root=/home/yoshihiro/mylib/jndidir
$
```

¹ Connector/J 5.1.28 に同梱されているガイド connector-j.pdf(revision: 36840 2013-11-27 版)

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

```
$ cat /home/yoshihiro/mylib/jndidir/mysqldsn.properties
type=javax.sql.DataSource
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost/mysqlDemoU?characterEncoding=utf8
user=yoshihiro
password=password
$
```

➔ 本検証は、UTF8 ロケール配下で試しています。そのため、URL エントリには「characterEncoding=utf8」を指定し、日本語を UTF8 にエンコードして取得できるように設定しています。

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

```
$ export CLASSPATH=$CLASSPATH:/home/yoshihiro/mylib/jndidir
$
```

4) MySQL サーバが起動中であることを確認

```
$ mysqladmin status -u yoshihiro -p
Enter password:
Uptime: 20288 Threads: 2 Questions: 371 Slow queries: 0 Opens: 102 Flush
tables: 1 Open tables: 68 Queries per second avg: 0.018
$
```

5) 検証用に作成したプログラムをコンパイル

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

6) プログラムを実行

```
$ java MYSQLTESTJ
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
$
```

- ➔ モジュールは javabyte で生成されているため、java コマンドを使って実行しています。
- ➔ 全て正常に処理できていることが確認できます。

以上