Micro Focus Visual COBOL 2.1J for x64/x86 Linux

Microsoft ODBC Driver 11 for SQL Server

動作検証 検証結果報告書

2013 年 7 月 24 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

Microsoft 社が提供する ODBC Driver 11 for SQL Server は Linux 上で実行されるネイティブアプリケーションより SQL Server 2008, SQL Server 2008 R2, 及び SQL Server 2012 への接続機能を提供する ODBC ドライバです。

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 ドライバを経由したリレーショナルデータベースアクセスを提供します。この OpenESQL は ODBC 3.0 準拠の ODBC ドライバとの連携を保証しており、ODBC Driver 11 for SQL Server は本稿執筆時点で上位 バージョンである ODBC 3.51 仕様に準拠しています。Microsoft が提唱する ODBC Driver の要件[1]が満たされていると仮定すれば OpenESQL を介して発行される ODBC 3.0 に準拠した ODBC 要求をドライバは正常に処理できると解釈できます。本稿では、これらの技術を利用し Linux 上の COBOL プログラムから Windows 上で稼働する SQL Server 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 プリプロセッサが搭載されています。弊社では製品出荷時に Oracle, SQL Server(Windows のクライアントからの呼び出し), DB2 については動作検証しています。この他の構成でも ANSI SQL 92 標準の SQL 文が COBOL プログラムに埋め込み SQL 文として書かれているのであれば OpenESQL を使って ODBC 3.x 仕様に準拠した ODBC ドライバを介してアクセスすることは設計上問題ありません。本検証では Windows 上で稼働する SQL Server に格納された日本語を含むデータを Linux マシンより ODBC Driver 11 for SQL Server を介してアクセスできることを検証しました。

OpenESQL は nchar 及び nvarchar で定義された列に対してクライアント側のロケールに関係なく一括して UCS2 の形式でデータ授受を処理する技術をサポートしています。この技術を利用して PIC N USAGE NATIONAL で定義したホスト変数を介して UTF8 ロケールだけでなく SJIS ロケール配下でも日本語を含むデータを正常に扱えることを確認しました。

---

[1] http://msdn.microsoft.com/en-us/library/ms712622(v=vs.85).aspx
Because all ODBC 3.x drivers are required to support the features in the Core interface conformance level, the following are true:
An ODBC 3.x driver will support all the features used by a standards-compliant application.

2. 検証環境

DB サーバ

■ ハードウェア

機種： Dell OPTIPLEX7010
CPU： Intel Core2 i7-3770 3.40GHz
Memory： 3.00 Gbyte memory(ゲスト OS に割り当てたサイズ)

■ ソフトウェア

Windows 8 Enterprise 64bit (VM のゲスト OS として稼働)
Microsoft SQL Server 2012 Standard Edition バージョン 11.0.3128.0

DB クライアント

■ 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)

■ ソフトウェア

Red Hat Enterprise Linux 6 Update 3(VM のゲスト OS として稼働)
Microsoft ODBC Driver 11 for SQL Server - RedHat Linux 11.0.2270.0
unixODBC ドライバマネージャ 2.3.0
Micro Focus Visual COBOL 2.1J Update 1

3. テスト内容

本検証で用意した COBOL プログラムには CREATE TABLE 文が埋め込み SQL 文として記述され、テスト用のテーブルを作成します。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、このデータを CURSOR – FETCH してデータ取り出しをします。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。また、扱うデータには日本語を交えます。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した。
UTF8 ロケール環境、SJIS ロケール環境ともに同一のプログラムを使って検証していますが、ソースにマルチバイト文字が含まれるため、それぞれのロケールに合ったかたちで

符号化したものを利用しています。

4. 結果

4.1 インストール

> Microsoft ODBC Driver 11 for SQL Server
以下のリンク先よりインストーラをダウンロードし、インストールしました。

ダウンロードファイル：msodbcsql-11.0.2270.0.tar.gz

ダウンロード元(2013/7/23 リンク検証)：
http://www.microsoft.com/en-us/download/details.aspx?id=36437

> unixODBC ドライバマネージャ
Microsoft ODBC Driver 11 for SQL Server のインストーラリソースに同梱されている unixODBC 2.3 インストール用のスクリプト build_dm.sh を流してインストールしました。

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。

① Windows 上で稼働する SQL Server 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 DISPLAY-OF 組み込み関数を使って UCS2 で符号化された取得データをロケールに応じた形式へ符号化
⑪ DROP TABLE 文にてテスト用に作成したテーブルを削除

⑫　SQL Server 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

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

Linux 版の Visual COBOL を使って開発されたCOBOL アプリケーションより Windows 上で稼働する SQL Server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。また、日本語データについてもロケールを問わず正常に扱えることも検証できました。

以上

付録. サンプルアプリケーションの実行

1) インストール時に更新された odbcinst.ini ファイルを確認します。

```
$ tail -7 /etc/odbcinst.ini

[ODBC Driver 11 for SQL Server]
Description=Microsoft ODBC Driver 11 for SQL Server
Driver=/opt/microsoft/msodbcsql/lib64/libmsodbcsql-11.0.so.2270.0
Threading=1
UsageCount=1

$
```

➔ 「ODBC Driver 11 for SQL Server」という名前でドライバが登録されていることを確認します。

2) 構成ファイル odbc.ini に Windows 上で稼働する SQL Server へアクセスするための DSN を設定します。この際、odbcinst.ini に埋め込まれたドライバ名を Driver エントリに指定します。

```
$ cat /etc/odbc.ini
[MSSQLTest]
Driver = ODBC Driver 11 for SQL Server
Server = tcp:10.18.12.139

$
```

3) unixODBC が提供する接続ツール isql をにて構成した DSN を使って Windows 上の SQL Server へ接続できることを確認します。

```
$ isql MSSQLTest coboltest password
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
```

```
| quit                                  |
|                                       |
+---------------------------------------+
SQL> SELECT local_net_address FROM sys.dm_exec_connections where session_id =
@@SPID
+-----------------------------------------------+
| local_net_address                             |
+-----------------------------------------------+
| 10.18.12.139                                  |
+-----------------------------------------------+
SQLRowCount returns 0
1 rows fetched
SQL>
```

➔ 正常に接続先のサーバ 10.18.12.139 を参照できていることが確認できます。

4) 検証用に作成したプログラムを UTF8 ロケール配下でコンパイルします。

```
$ locale
LANG=ja_JP.UTF-8
LC_CTYPE="ja_JP.UTF-8"
LC_NUMERIC="ja_JP.UTF-8"
LC_TIME="ja_JP.UTF-8"
LC_COLLATE="ja_JP.UTF-8"
LC_MONETARY="ja_JP.UTF-8"
LC_MESSAGES="ja_JP.UTF-8"
LC_PAPER="ja_JP.UTF-8"
LC_NAME="ja_JP.UTF-8"
LC_ADDRESS="ja_JP.UTF-8"
LC_TELEPHONE="ja_JP.UTF-8"
LC_MEASUREMENT="ja_JP.UTF-8"
LC_IDENTIFICATION="ja_JP.UTF-8"
LC_ALL=
$ cob -u sqlsrvtest.cbl -C"SOURCEFORMAT(VARIABLE) SQL(DBMAN==ODBC ODBCV3)"
$
```

5) プログラムを実行します。

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

Enter data source (e.g. odbcdemo) MSSQLTest
Enter username[('.'|'/')password] (e.g. scott/tiger) coboltest/password
Create table
Insert first row
Insert second row containing Japanese characters
Commit the insertion
Update row
Commit the change
Fetch
************************************
int_col     : +00001
nvarchar_col: alphanumeric value
************************************
int_col     : +00002
nvarchar_col: たちつてと
************************************
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。

6) 同様のテストを SJIS ロケール配下で実施します。

```
$ locale
LANG=ja_JP.sjis
LC_CTYPE="ja_JP.sjis"
LC_NUMERIC="ja_JP.sjis"
LC_TIME="ja_JP.sjis"
LC_COLLATE="ja_JP.sjis"
LC_MONETARY="ja_JP.sjis"
LC_MESSAGES="ja_JP.sjis"
```

```
LC_PAPER="ja_JP.sjis"
LC_NAME="ja_JP.sjis"
LC_ADDRESS="ja_JP.sjis"
LC_TELEPHONE="ja_JP.sjis"
LC_MEASUREMENT="ja_JP.sjis"
LC_IDENTIFICATION="ja_JP.sjis"
LC_ALL=
$ cob -u sqlsrvtest.cbl -C"SOURCEFORMAT(VARIABLE) SQL(DBMAN==ODBC ODBCV3)"
$ cobrun sqlsrvtest.gnt
Create/insert/update/select/drop test

Enter data source (e.g. odbcdemo) MSSQLTest
Enter username[('.'|'/')password] (e.g. scott/tiger) coboltest/password
Create table
Insert first row
Insert second row containing Japanese characters
Commit the insertion
Update row
Commit the change
Fetch
************************************
int_col     : +00001
nvarchar_col: alphanumeric value
************************************
int_col     : +00002
nvarchar_col: たちつてと
************************************
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