Visual COBOL 7.0J for x64/86 Linux

Amazon Aurora PostgreSQL
動作検証 検証結果報告書

2022 年 5 月 25 日作成

昨今、様々な業界でクラウドを利用したシステムの開発・運用が進められており、COBOL で開発された基幹システムにおいても、クラウド利用の波は押し寄せています。

クラウドを利用することによるハードウェアコストの削減だけではなく、クラウドが提供する各種サービスの活用も利点の１つです。例えば、マネージド型のデータベースサービスは、容易に、データベースの稼働・バックアップなどが行えるサービスであり、クラウド上で稼働するシステムで多く利用されています。

では、データベース連携を伴う COBOL で開発されたシステムのクラウド利用を考える際、このようなマネージド型のデータベースサービスを利用できるのでしょうか。実は、マネージド型のデータベースサービスとは、ODBC, JDBC といった標準仕様に基づいた接続が行えるため、COBOL で開発されたシステムでも、これまでオンプレミス環境で利用してきた接続方式で、マネージド型のデータベースサービスを利用することができます。

この文書は、Micro Focus Visual COBOL が提供する各種データベース接続機能を利用し、マネージド型のデータベースサービスの１つである、Amazon Web Services (AWS) が提供する Amazon Aurora との接続性を検証した結果報告書です。

目次

1. 検証概要、目的及びテスト方法

1) 検証概要

Amazon Aurora は、AWS (Amazon Web Services) が開発したフルマネージド型のリレーショナルデータベースエンジンです。昨今のトレンドである各種クラウドサービスを活用したシステム構築・運用によって、Amazon Aurora を用いた業務システムの運用も増加しています。

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 ドライバー、JDBC ドライバー、ADO.NET データプロバイダー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。このリレーショナルデータベースのアクセス先には、既存のデータベース製品だけではなく、Amazon Aurora といったクラウドが提供するマネージドデータベースサービスも含まれます。

Linux 環境のシステムロケールが UTF-8 であっても、Visual COBOL を利用頂くことで Shift_JIS で記述された COBOL アプリケーションを実行することができ、また、これらのアプリケーションと Amazon Aurora との連携も可能になります。このことについては弊社ホームページで公開しているホワイトペーパー「Linux 標準ロケールで COBOL を利用する」にて紹介していますように、このような環境においても、Visual COBOL を利用頂くことで Shift_JIS で記述された COBOL アプリケーションを実行することができ、また、そのようなアプリケーションと Amazon Aurora との連携も可能になります。

本稿では、この OpenESQL を使って ODBC 並びに JDBC 経由で、Shift_JIS で記述された埋め込み SQL 文を含む COBOL プログラムから Amazon Aurora PostgreSQL へアクセスできることを動作検証しました。

2) 目的及びテスト方法

Micro Focus Visual COBOL は最新鋭の COBOL 言語開発・実行環境を提供しています。COBOL 言語への埋め込み SQL 処理系を標準装備しており、ODBC ドライバー、JDBC ドライバー、ADO.NET データプロバイダー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ています。OpenESQL を使うことで、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対して設計上問題なくアクセスすることができます。本稿執筆時点における Micro Focus Visual COBOL の最新版　7.0J で動作保証されている PostgreSQL 11.15 互換の Amazon Aurora PostgreSQL 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 のネイティブ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、Java バイトコードにコンパイルし、JVM クラスとして実行して動作を確認しています。

2. 検証環境

＞ EC2 インスタンス (Linux クライアント) 環境

OS	Red Hat Enterprise Linux
インスタンスタイプ	t2.micro (1 vCPU, 1GiB)
COBOL 開発環境製品	Micro Focus Visual COBOL 7.0J for x64/86 Red Hat Patch Update 7
JDK/JRE	Adopt OpenJDK 11.0.15+10
ソフトウェア	postgresql-odbc 10.03.0000-2.el8 Simple-JNDI 0.11.4.1 PostgreSQL JDBC ドライバー 42.3.5 unixODBC 2.3.7-1 Shift_JIS 環境用に、以下を導入しています。 glibc-locale-source-2.28-189.1

＞ RDS (Amazon Aurora PostgreSQL)

インスタンスクラス	Db.r6g.large
エンジンバージョン	11.15
ネットワーク	パブリックアクセスなし EC2 環境から PostgreSQL 通信のみ許可

3. テスト内容

以下の内容が実施したテストの概要です。

1) ODBC 接続 (ユーザーロケール Shift_JIS)

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を作成します。続いて、INSERT 文によるデータの登録、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、データを確認します。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。こちらの検証は、ユーザーロケール Shift_JIS を設定した環境で実施します。

2) ODBC 接続 (COBUTF8 利用)

デフォルトのシステムロケールである UTF-8 環境下で COBUTF8 を利用することで、ユーザーロケールを切り替えることなく、1) で利用するプログラムが正常に処理できることを検証します。

3) JDBC 接続

1) で利用するプログラムソースの接続方式を JDBC に変更したプログラムを利用し、JDBC 接続上でも、同じロジックが正常に処理できることを検証します。

4. 結果

1) インストール

- Amazon Aurora PostgreSQL

  今回は動作検証が目的であるため、以下のように、「開発/テスト」テンプレートを選択して構築しました。

- unixODBC

  以下のコマンドを実行し、インストールしました。

  yum install unixODBC

- postgresql-odbc (PostgreSQL ODBC ドライバー)

  以下のコマンドを実行し、インストールしました。

  yum install postgresql-odbc

- PostgreSQL JDBC ドライバー

  以下のリンク先よりソースをダウンロードし、インストールしました。

  ダウンロード元(2022/5/19 リンク検証済) :

  https://jdbc.postgresql.org/download.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(2022/5/19 リンク検証済) :

  https://mvnrepository.com/artifact/simple-jndi/simple-jndi/0.11.4.1

- Shift_JIS リソース設定

  以下のコマンドを実行し、インストールしました。

  yum install glibc-locale-source

  localedef --no-warnings=ascii -i ja_JP -f SHIFT_JIS ja_JP.SJIS

2) サンプルアプリケーションの作成

本検証で用意したプログラムの処理フローを以下に記します。実際のプログラムコードを、Micro Focus のウェブサイト上に本報告書と供に公開しています。

- ODBC および JDBC 接続検証に使用したプログラムフロー
  - ① Amazon Aurora 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 Amazon Aurora PostgreSQL データベースとの接続を切断

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。検証の実行手順等の詳細は付録の通りとなります。

5. テスト結果及び考察

Visual COBOL を使用してネイティブコードにコンパイルした、埋め込み SQL 文を含む COBOL プログラムから ODBC 経由で Amazon Aurora PostgreSQL データベースに接続し,DDL 文、DML 文、DCL 文を発行してデータベースが操作できることを検証しました。なお、本検証は、ユーザーロケールに Shift_JIS を設定する方法と、COBUTF8 を利用する 2 つの方法で実施しました。

一方、Java バイト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。

ネイティブコード、Java バイトコード何れの検証においても、日本語を含んだデータについて正常に扱えることも検証できました。

以上

6. 付録 サンプルアプリケーション実行結果

1) ODBC 接続 (ユーザーロケール Shift_JIS)

I. unixODBC の構成ファイルに Amazon Aurora PostgreSQL へのアクセス情報を設定

```
$ cat /etc/odbcinst.ini
[PostgreSQL]
Description     = ODBC for PostgreSQL
Driver          = /usr/lib64/psqlodbcw.so
FileUsage       = 1
・・
$ cat /etc/odbc.ini
[AuroraPostgreSQLODBC]
Driver=PostgreSQL
SERVERNAME=AmazonAuroraPostgreSQL.ap-northeast-
1.rds.amazonaws.com
PORT=5432
USERNAME=postgres
PASSWORD=postgresSQLPassword
DATABASE=vcdb
```

II. ODBC 経由で Amazon Aurora PostgreSQL への接続確認

```
$ isql   AuroraPostgreSQLODBC
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL> select current_timestamp;
```

```
+--------------------------+
| current_timestamp            |
+--------------------------+
| 2022-05-25 01:16:34.192519|
+--------------------------+
SQLRowCount returns 1
1 rows fetched
SQL> quit
$
```

III. Shift_JIS 環境の確認

```
$ echo $LANG
ja_JP.sjis
$ echo "あ" |od -x
0000000 a082 000a
0000003
$
```

Shift_JIS のコードである 82A0 が戻されていることが確認できます。

IV. 検証用に作成したプログラムをコンパイル

```
$ cob -u PSQLTESTO.cbl
$
```

V. プログラムを実行

```
$ cobrun PSQLTESTO.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
$
```

全て正常に処理されていることが確認できます。

2) ODBC 接続 (COBUTF8 利用)

ODBC 設定情報は、1) と同様です。

I. ユーザーロケールが システムデフォルト UTF-8 であることの確認

```
$ echo $LANG
ja_JP.utf8
$ echo "あ" |od -x
0000000 81e3 0a82
0000004
$
```

UTF-8 のコードである E38182 が戻されていることが確認できます。

II. 検証用に作成したプログラムをコンパイル

```
$ cobutf8 cob -u PSQLTESTO.cbl
$
```

III. プログラムの実行

```
$ cobutf8 cobrun PSQLTESTO.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
$
```

全て正常に処理されていることが確認できます。

3) JDBC 接続

I. Java のバージョンを確認

```
$ java -version
openjdk version "11.0.15" 2022-04-19
OpenJDK Runtime Environment Temurin-11.0.15+10 (build
11.0.15+10)
OpenJDK 64-Bit Server VM Temurin-11.0.15+10 (build 11.0.15+10,
mixed mode)
```

II. ダウンロードした JDBC ドライバー と Simple-JNDI のライブラリを CLASSPATH に追加

```
$ export CLASSPATH=javalib/postgresql-42.3.5.jar:javalib/simple-
jndi-0.11.4.1.jar:SimpleJNDI:$CLASSPATH
$
```

III. Simple-JNDI の設定ファイルを編集

jndi.properties

```
$ cat jndi.properties
java.naming.factory.initial=org.osjava.sj.SimpleContextFactory
org.osjava.sj.root=/home/ec2-
user/work/jvm/SimpleJNDI/AurorapostgreSQL
```

vcdb.properties

```
$ cat AurorapostgreSQL/vcdb.properties
type=javax.sql.DataSource
url=jdbc:postgresql://AmazonAuroraPostgreSQL.ap-northeast-
1.rds.amazonaws.com:5432/vcdb
driver=org.postgresql.Driver
user=postgres
password=postgresPassword
```

IV. 検証に使用するプログラムのコンパイル

```
$ cob -C"jvmgen(main)" PSQLTESTJ.cbl
$
```

V. 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
```

全て正常に処理されていることが確認できます。

以上