Micro Focus Visual COBOL チュートリアル

OpenESQL アシスタントを利用したデータベースアクセス

Visual Studio 編

1. 目的

レガシーな COBOL 言語で開発された企業システムからデータベースのアクセスを行うには繁雑な技術を要するというイメージをお持ちの方も多いかもしれませんが、Visual COBOL を使えば驚くほど簡単に開発が可能です。一般的にリレーショナル・データベースのデータ操作には SQL を利用します。SQL は、COBOL と異なるデータベース言語となるため、COBOL コンパイラは SQL 文を解釈できません。
そこで、COBOL 上で SQL 文を利用する場合、EXEC SQL と END-EXEC で囲んだ部分のみに SQL 文を記述し、COBOL と区別させます。この SQL 文が埋め込まれたソースを Oracle などデータベースベンダが提供するプリコンパイラに渡すことで、SQL 文の部分をデータベースベンダが提供する API コールに展開した COBOL プログラムを生成します。プログラマーが当初作成するソースはこのソースではなくプリコンパイル展開する前のソースになるためデバッグ時のコードと作成時のコードに相違が生まれてしまいます。

Visual COBOL はプログラマーが実際にメンテナンスする埋め込み SQL 文が入ったソースを IDE に認識させるべく、プリコンパイルとコンパイルをシングルステップで処理できるようにしました。プリコンパイル、もしくはそれに相当する処理を Visual COBOL が内部的に処理します。これにより、プリコンパイル後のソースは扱うことがなくなるため、このソースに関する考慮は不要となりました。

Visual COBOL はこのプリコンパイルを意識させない技術に関して技術や利用する DBMS に応じて柔軟に技術選択いただけるよういくつかのオプションを用意しています。その中でも今回紹介する OpenESQL は、マイクロフォーカス純正のプリプロセッサです。また、OpenESQL によるデータベースアクセスを行う場合、「OpenESQL アシスタント」という開発補助ユーティリティを利用できます。

このチュートリアルでは、OpenESQL アシスタントを使用して OpenESQL によるデータベースアクセスの方法を学びます。

2. 前提条件

本チュートリアルは、下記の環境を前提に作成されています。

- データベースサーバー

OS	Windows Server 2019 Standard Edition（64bit）
DBMS 製品	SQL Server 2019 Standard Edition

- 開発クライアント ソフトウェア

OS	Windows 11 Business Edition（64bit）
COBOL 製品	Visual COBOL 8.0 for Visual Studio 2022
DBMS 製品	SQL Server Native Client 11.0

- チュートリアル用サンプルプログラム

  下記のリンクから事前にチュートリアル用のサンプルデータベースの SQL ファイルをダウンロードして、任意のフォルダに解凍しておいてください。

  サンプルデータベースの SQL スクリプトダウンロード

内容

3. チュートリアル手順の概要

3.1. データベースの準備

1) チュートリアル用データベースの作成

チュートリアルで使用するデータベースとサンプルテーブル、データのインポートを行う SQL を実行します。

① SQL Server Management Studio の実行

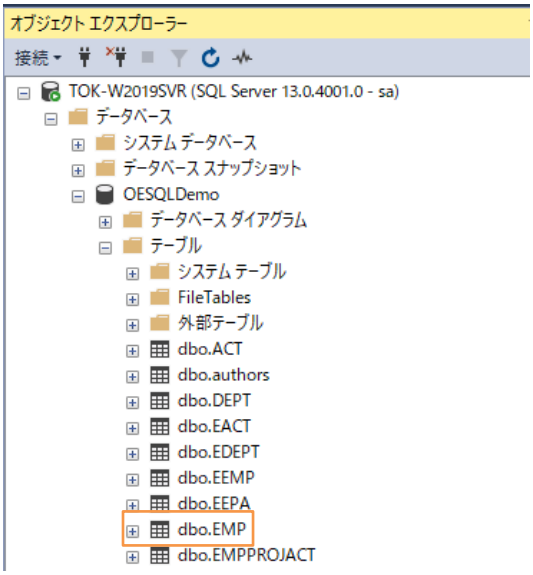
Microsoft SQL Server Management Studio を起動し、管理者でログインを行います。※Microsoft SQL Server Management Studio は別途ダウンロードが必要です。

② SQL スクリプトの実行

ダウンロードした SQL スクリプトの内容をコピー/ペーストしてクエリーを実行します。

③ 実行結果の確認

初めて実行する場合はデータベースが存在しないので drop database 文だけが失敗します。[オブジェクトエクスプローラー] ツリーにてデータベースとして「OESQLDemo」が作成されており、EMP テーブルが存在することを確認します。



④ SQL Server Management Studio の終了

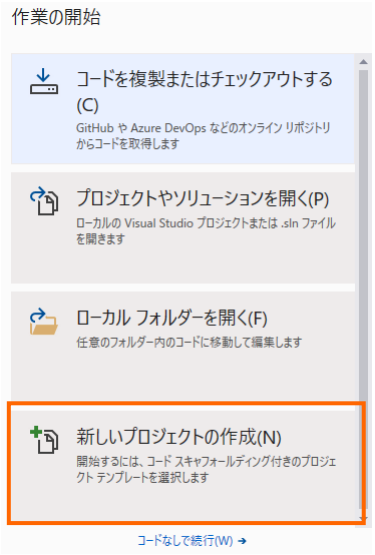
閉じるボタンにて「Microsoft SQL Server Management Studio」を終了します。

3.2. Visual Studio プロジェクトの作成と設定

1) Visual Studio の起動とプロジェクトの作成

① Visual Studio XXXX (XXXX はバージョン番号) を起動します。※今回は Visual Studio 2022 を起動します。

② 「新しいプロジェクトの作成(N)」を選択します。

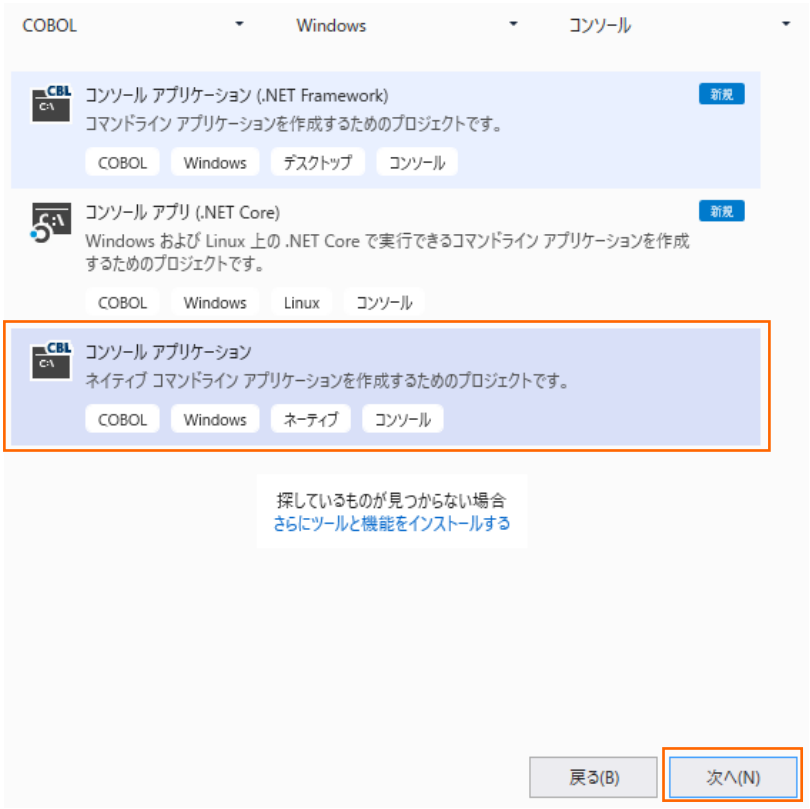


③ 以下のフィルタを設定し、[コンソールアプリケーション] を選択し、[次へ(N)] をクリックします。

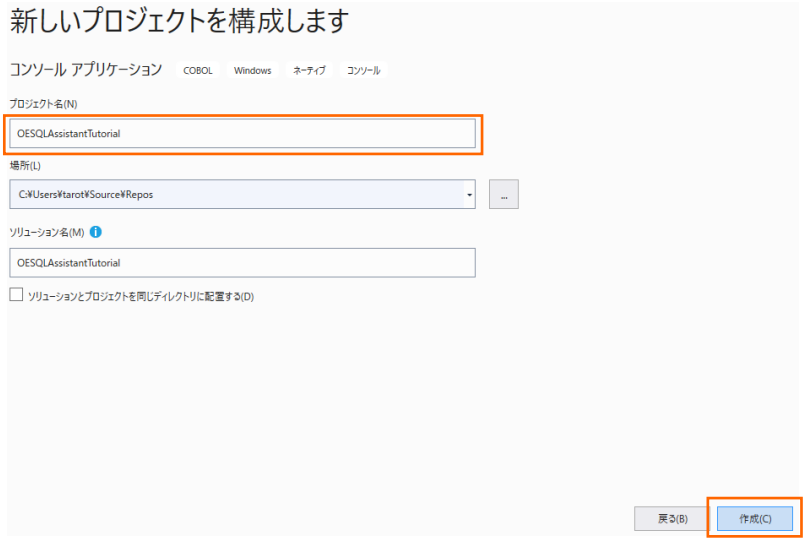
全ての言語： COBOL

全てのプラットフォーム： Windows

全てのプロジェクトの種類： コンソール

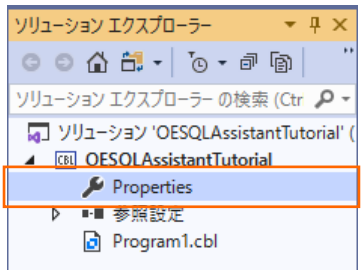


④ プロジェクト名に “OESQLAssistantTutorial” を入力し、[作成(C)] をクリックします。

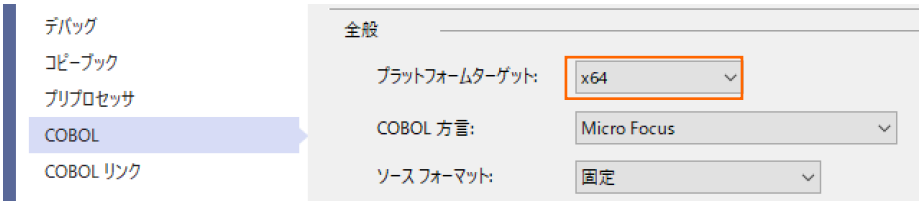


2) OpenESQL のプロジェクト設定

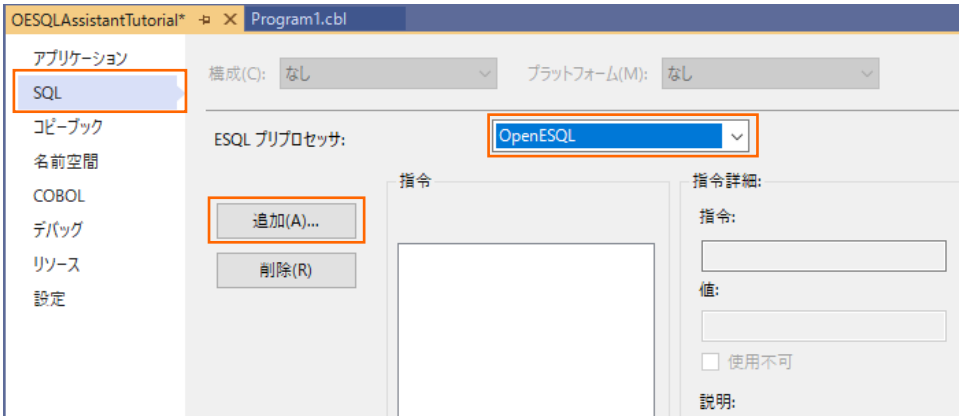
① OESQLAssistantTutorial プロジェクト配下の「Properties」を選択し、ダブルクリックします。



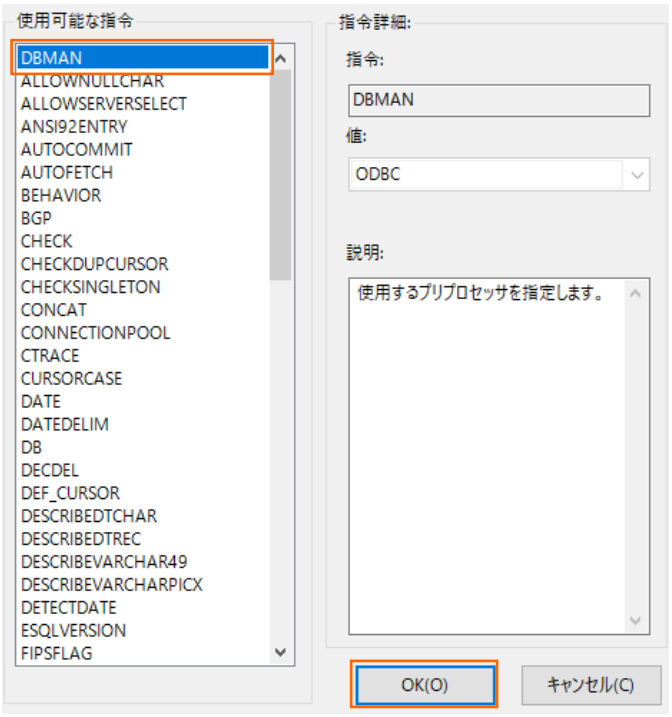
② [COBOL] タブを開き、[プラットフォームターゲット] に “x64” に変更します。※Visual Studio2019 にて ODBC 32 ビットデータソースを使用する場合、この操作は必要ありません。



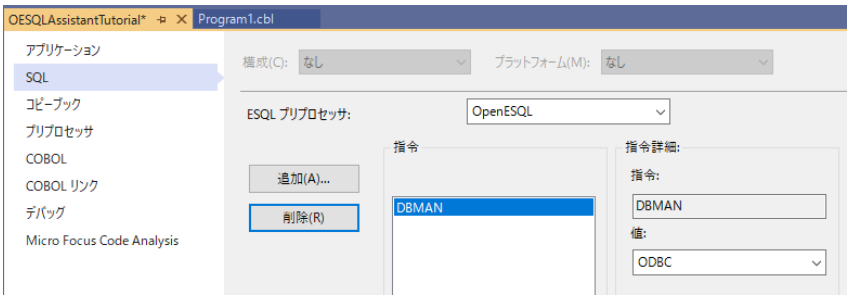
③ 次に [SQL] タブを開き、[ESQL プリプロセッサ] に “OpenESQL” を選択した上で、[追加(A)] をクリックします。



④ 「使用可能な指令」欄より “DBMAN” を選択し、[OK] をクリックします。

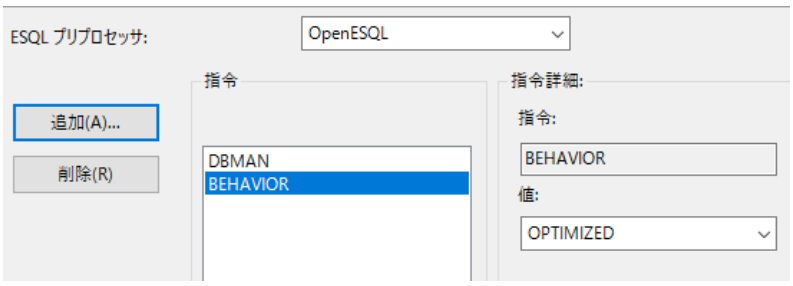


DBMAN が追加されます。

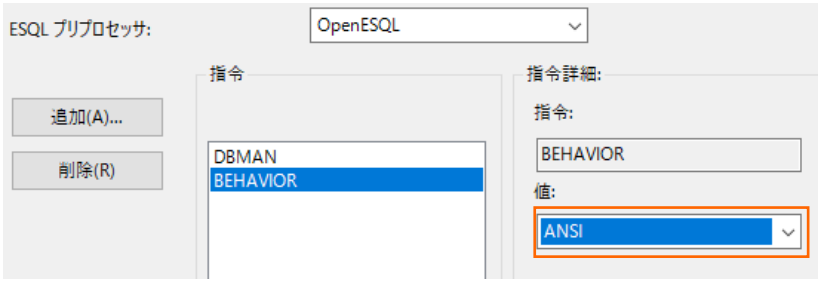


⑤ ⑥、⑦ の手順を再度実施し、”BEHAVIOR” を追加します。

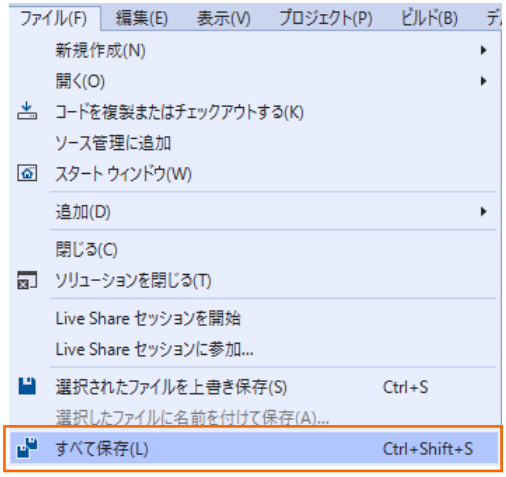
追加後、以下のように 2 つの指令が表示されます。



⑥ 「指令」欄より「BEHAVIOR」を選択し、右側の「値」を “ANSI” に変更します。

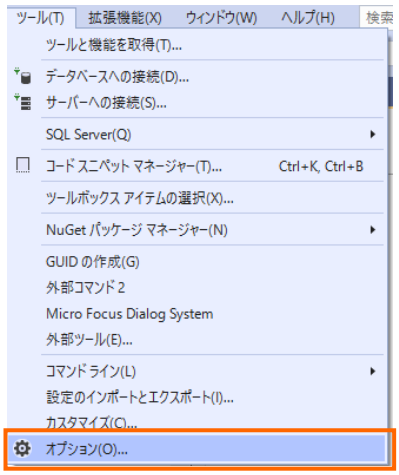


⑦ IDE メニューの [ファイル(F)] > [すべて保存(L)] を選択し、変更を保存します。



3) OpenESQL の設定変更

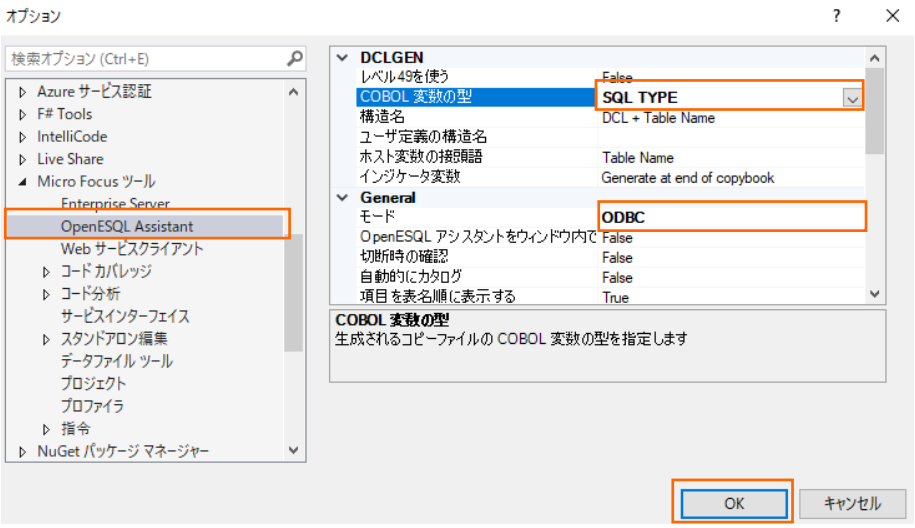
① Visual Studio のメニューより、[ツール(T)] > [オプション(O)] を選択します。



② ダイアログ左側より「Micro Focus ツール」配下の「OpenESQL Assistant」を選択し、以下の設定を行ったうえで、[OK] をクリックします。

COBOL 変数の型： SQL TYPE

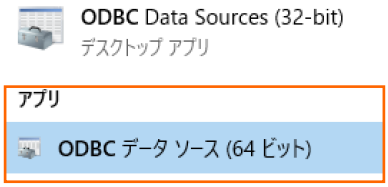
モード： ODBC



3.3. ODBC データソースの設定

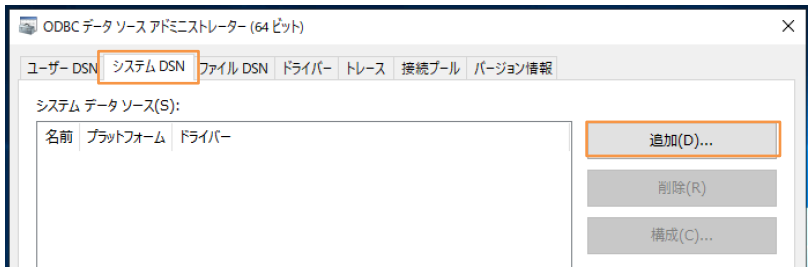
1) ODBC データソースアドミニストレーター（64 ビット）の起動 ※Visual Studio 2019 を使用する場合、32 ビットの ODBC データソースを作成してください。

① スタートメニュー横の検索にて ”ODBC” とタイプし、検索候補から「ODBC データソース（64 ビット）」を起動します。

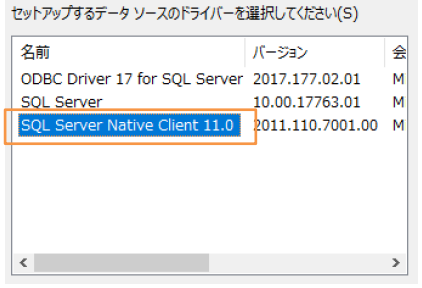


2) データソースの設定と接続確認

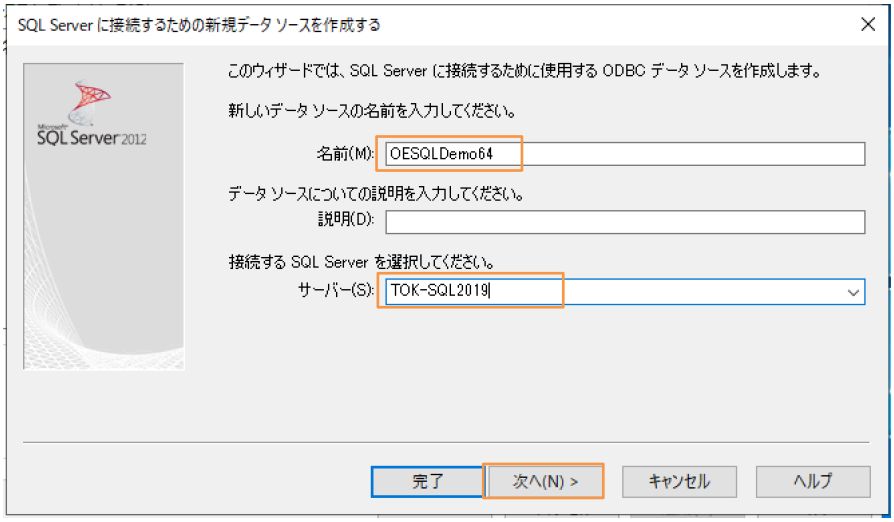
① ODBC データソースアドミニストレーター（32 ビット）画面より [システム DSN] タブをクリックし、[追加(D)…] ボタンをクリックします。



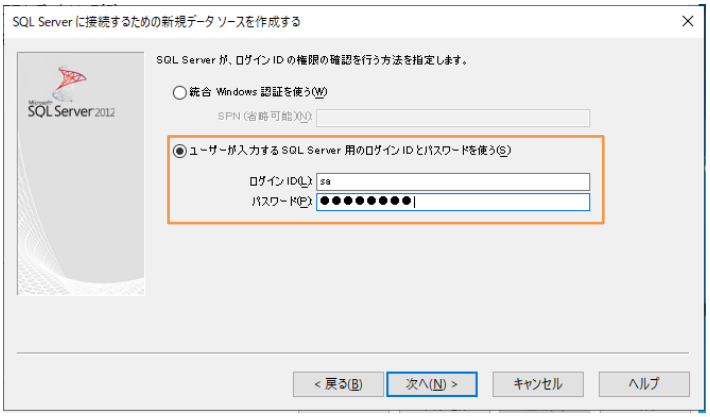
② [データソースの新規作成] ダイアログが表示されるので「SQL Server Native Client 11.0」を選択し、[完了] ボタンをクリックします。



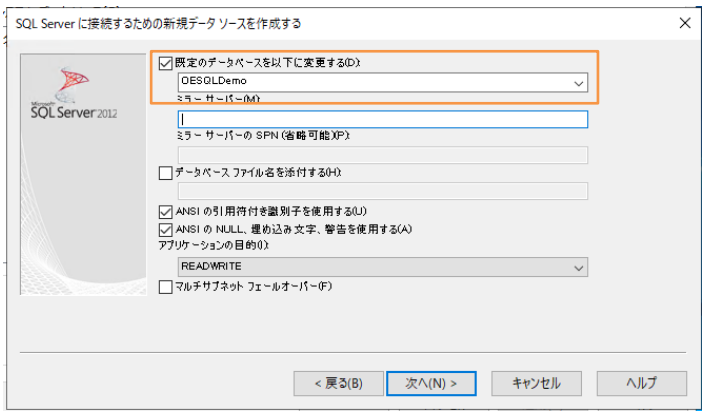
③ [Name] フィールドは、”OESQLDemo64” と入力し、[Server] フィールドは接続する SQL Server 名を指定し、[次へ(N)] ボタンをクリックします。もし、不明な場合は、自社の DBMS 管理者に確認してください。



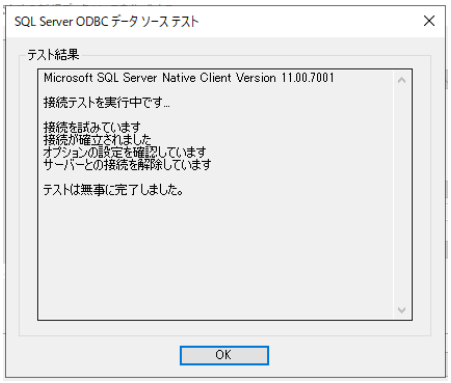
④ 設定されている認証方式を選択し [次へ(N)] ボタンをクリックします。不明な場合は、自社の DBMS 管理者に確認してください。（この例では混合モード認証で設定しています）



⑤ [既定のデータベースを以下に変更する(D):] にチェックを入れて、作成したデータベース「OESQLDemo」を指定し、[次へ(N)] ボタンをクリックします。



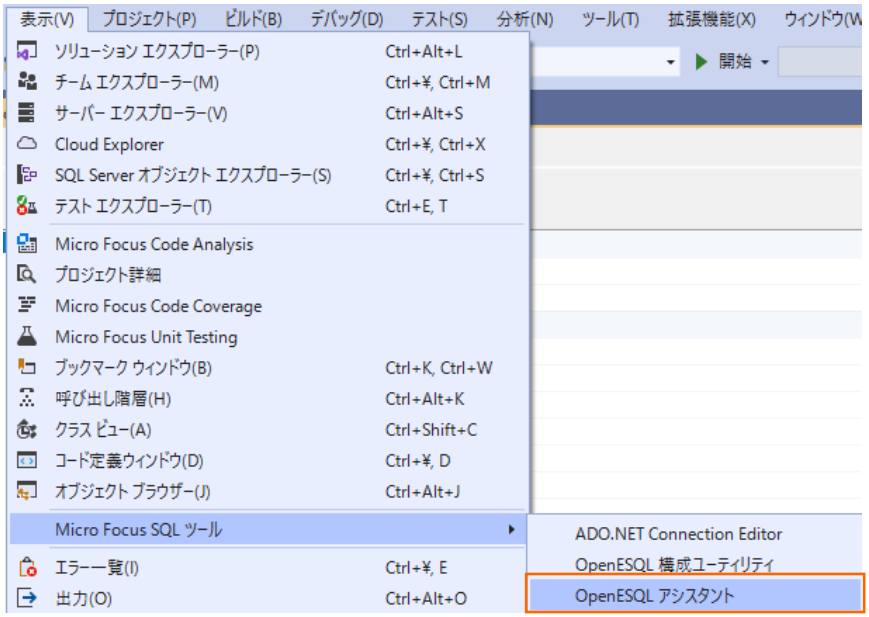
⑥ [完了] ボタンをクリックして設定を保存します。[データソースのテスト(T)] をクリックして正常に接続できるか確認してください。



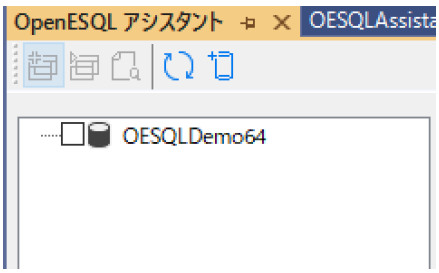
⑦ [OK] ボタンをクリックすると ODBC データソースアドミニストレーター（64 ビット）画面に戻ります。

3.4. クエリーの作成とテスト

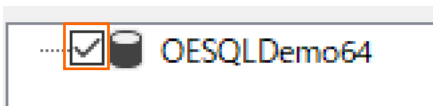
① IDE のメニューより [表示(V)] > [Micro Focus SQL ツール] > [OpenESQL アシスタント] を選択します。



OpenESQL アシスタントが開き、「OESQLDemo64」が一覧に表示されていることを確認してください。

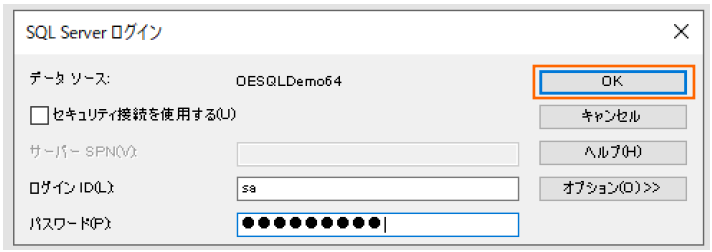


② OpenESQL アシスタント上の「OESQLDemo64」にチェックを行います。

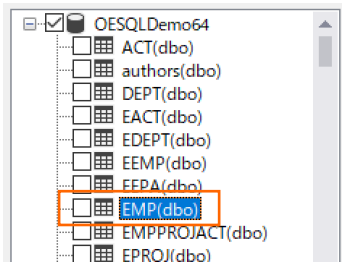


チェックした際に、以下のダイアログが表示された場合は、「OESQLDemo64」を選択し、[OK] をクリックします。

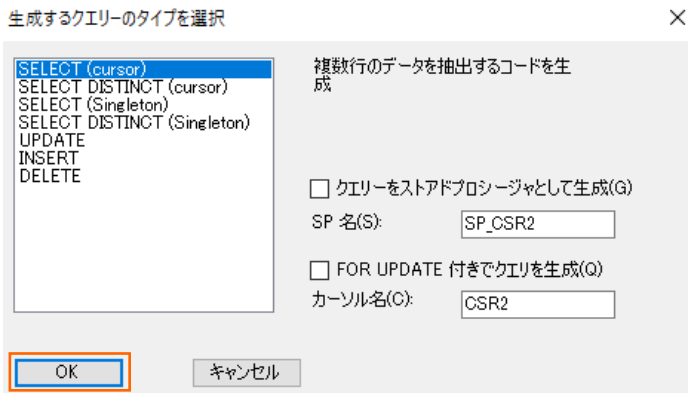
SQL Server へのログイン情報を求められた場合は入力した上で、[OK] をクリックします。



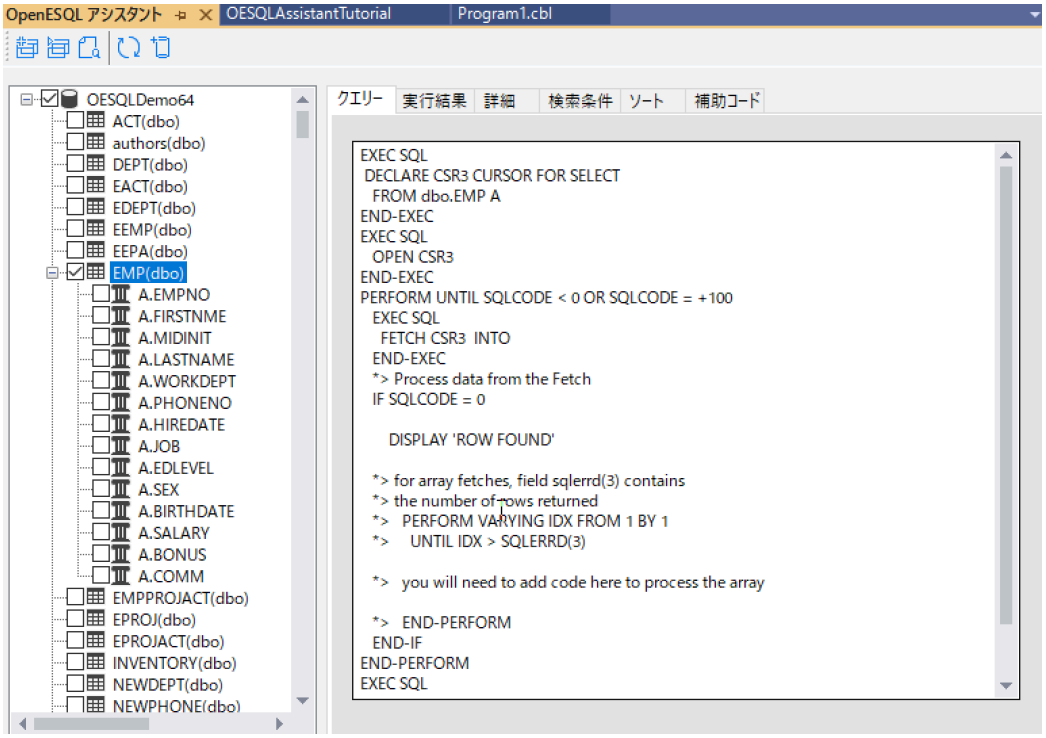
③ 「EMP(dbo)」にチェックを行います。



表示されたダイアログから作成するクエリーを選択できます。今回は、「SELECT (cursor)」 を選択し、[OK] をクリックします。



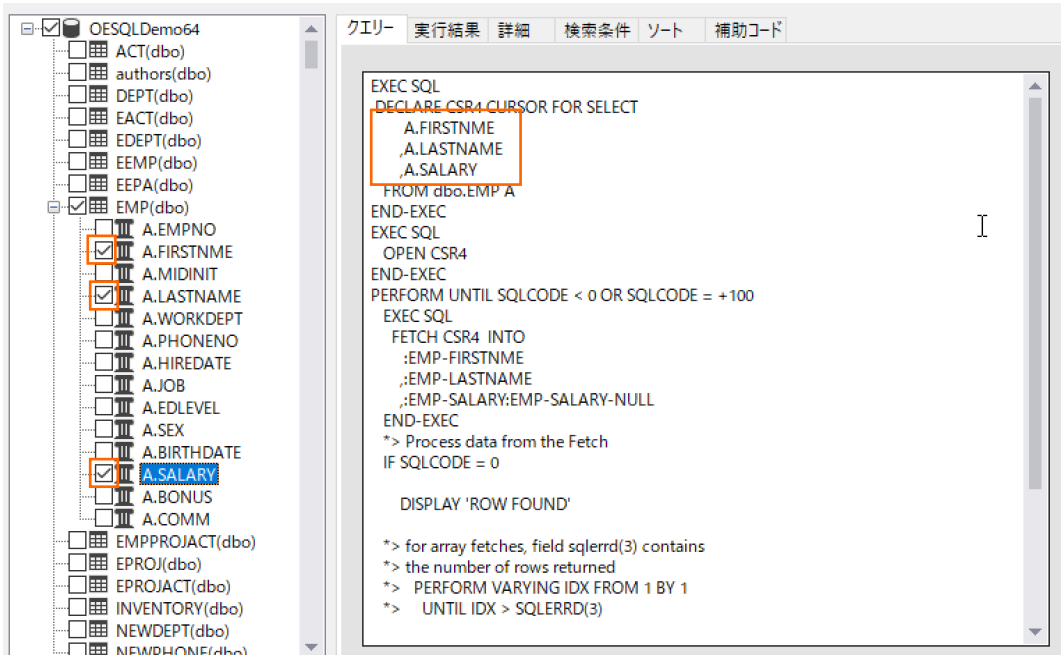
EMP テーブルの項目一覧が表示され、右側にクエリーのテンプレートが表示されます。



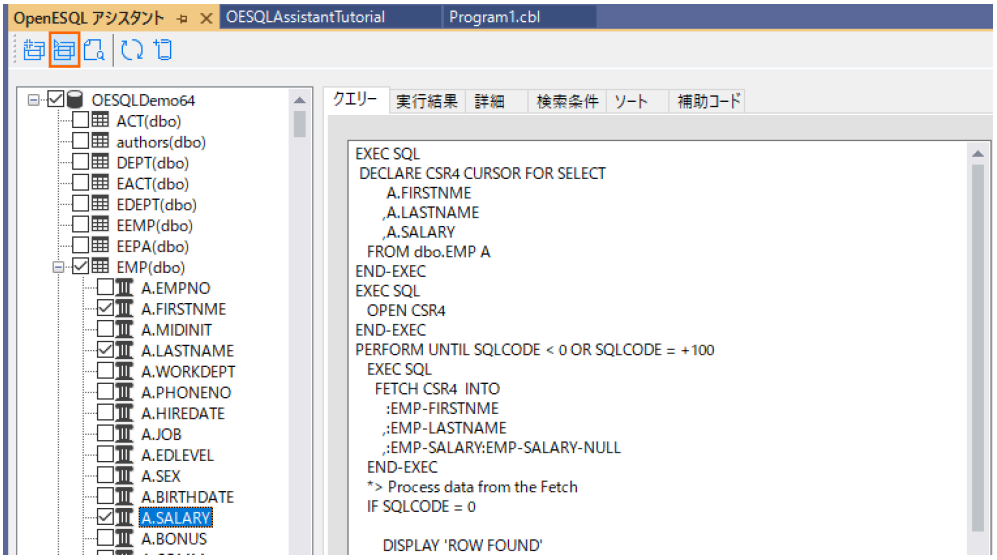
テーブル項目名が “A.” から始まっていますが、これはクエリーの 3 行目で EMP テーブルのエイリアスを “A” としているためです。複数テーブルを JOIN する場合、エイリアスが “B”, “C”, … と設定することができ、異なるテーブルに同名の項目が存在した場合でも、エイリアスを利用することで正しく取得先を識別することができます。

④ 「A.FIRSTNME」、「A.LASTNAME」、「A.SALARY」 をチェックします。

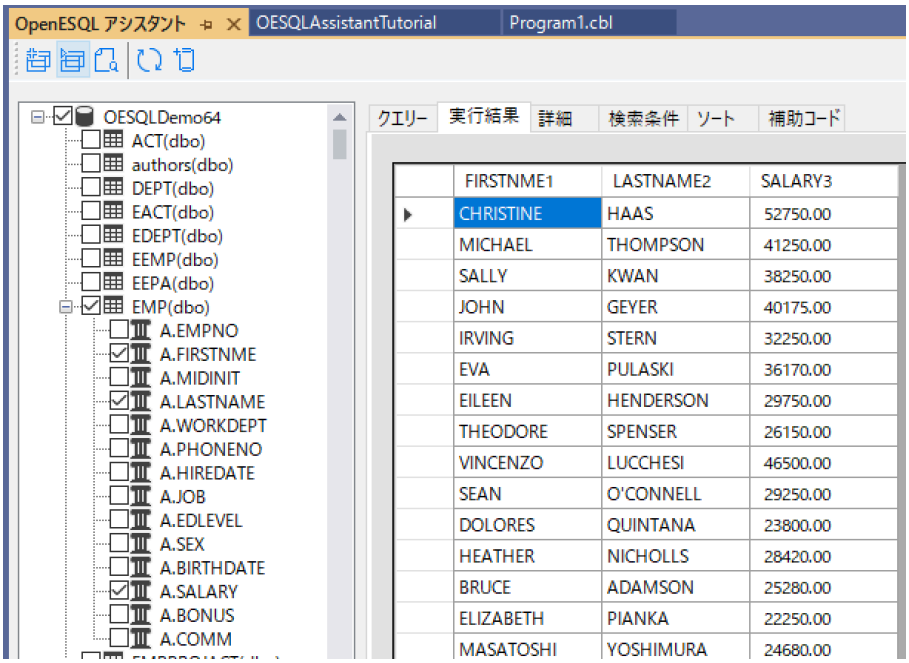
チェックを行うことで、SELECT 項目に追加されていることを確認します。



2) 「クエリーを実行する」アイコンをクリックして、プログラムを実行します。



「実行結果」タブに切り替わり、検索結果が表形式で表示されます。

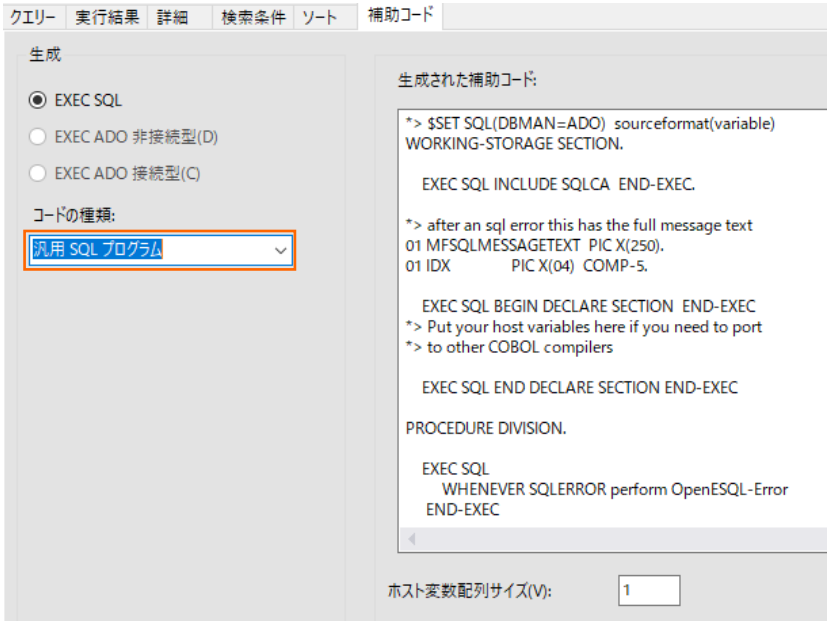


3.5. SQL 埋め込みプログラムの作成

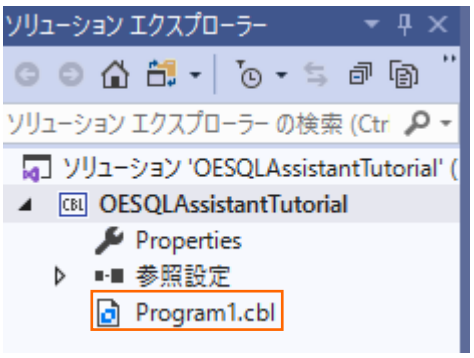
1) OpenESQL アシスタントを利用した埋め込みプログラムの作成

① OpenESQL アシスタント上の「補助コード」タブを選択します。

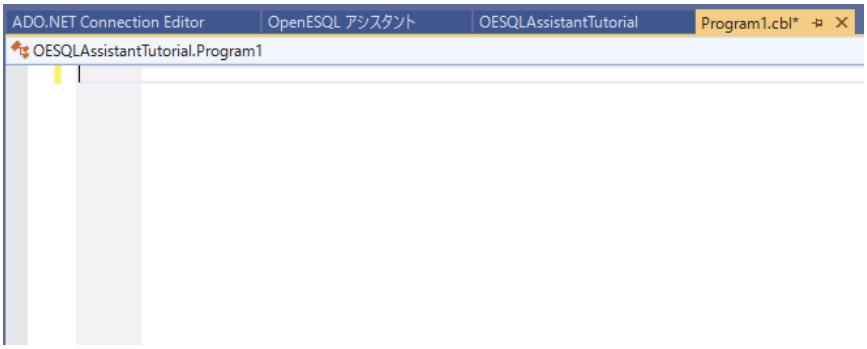
② 「コードの種類」に “汎用 SQL プログラム” を選択します。



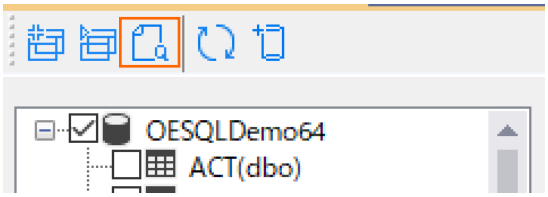
③ ソリューションエクスプローラーより、OESQLAssitantTutorial プロジェクト内の「Program1.cbl」をダブルクリックして開きます。



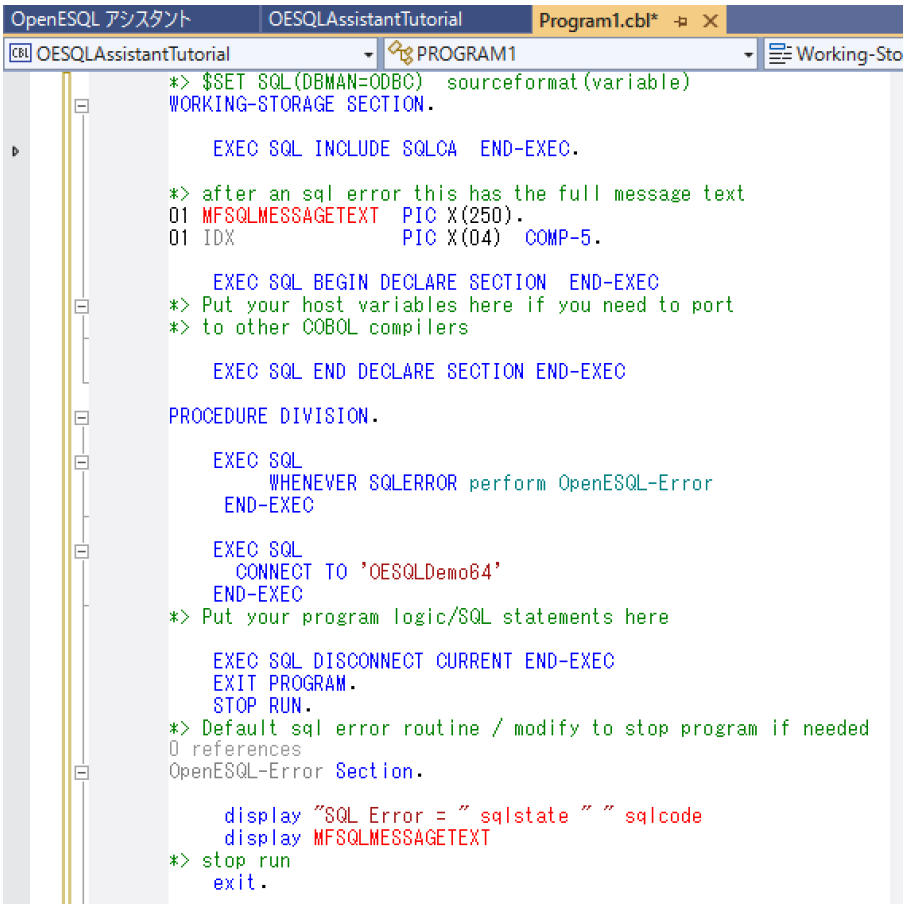
Program1.cbl の既存コードを削除してください。



④ OpenESQL アシスタントに戻り、「現在のクエリーにプログラムを挿入する」アイコンをクリックします。



さきほどの「Program1.cbl」に戻ると、プログラムが挿入されています。



⑤ IDE のメニューより [ファイル(F)] > [Program1.cbl の保存(S)] を選択して、変更を保存します。

⑥ Program1.cbl の “CONNECT TO 'OESQLDemo’” の末尾に “USER ‘データベースユーザーID.パスワード’ ” を記述します。以下の例では、データベースユーザーID に "sa", パスワードに "HogeHoge"を指定しています。

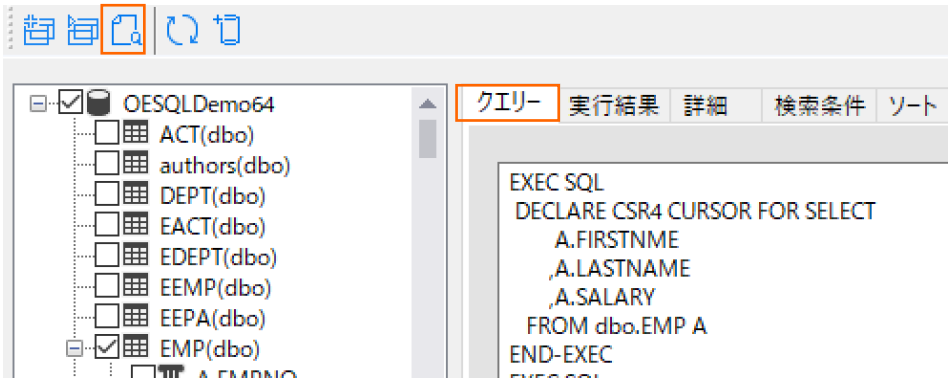
```
EXEC SQL
  CONNECT TO 'OESQLDemo64' USER 'sa.hogehoge'
END-EXEC
```

補足）
上記例は、SQL Server 認証が有効な場合の記述例です。環境に合わせ、データベースユーザーID にはデータベースにアクセス可能なログインユーザー、 パスワードにはデータベースユーザーID のパスワードを設定してください。なお、認証方式や情報については、自社のデータベース管理者にお問い合わせください。

⑦ Program1.cbl の “Put your program logic/SQL statements here” の次の空行をクリックします。

```
*> Put your program logic/SQL statements here

    EXEC SQL DISCONNECT CURRENT END-EXEC
    EXIT PROGRAM.
    STOP RUN.
*> Default sql error routine / modify to stop program if needed
0 references
OpenESQL-Error Section.

     display "SQL Error = " sqlstate " " sqlcode
     display MFSQLMESSAGETEXT
*> stop run
    exit.
```

⑧ 再度、OpenESQL アシスタントを開き、「クエリー」 タブを選択の上、[現在のクエリーにプログラムを挿入する] をクリックします。

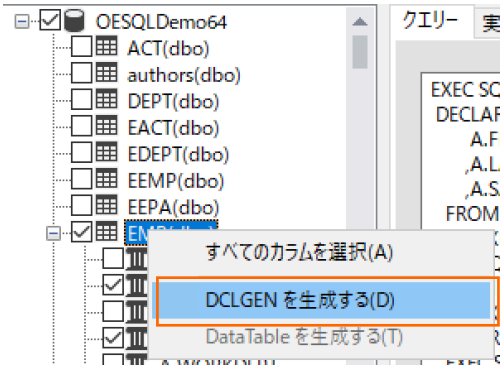


再度、「Program1.cbl」 に戻ると、新たにコードが追加されています。現時点ではホスト変数項目が定義されていないため、エラーとなっていますが、次の手順で対応します。ここでは、この状態で変更を保存します。

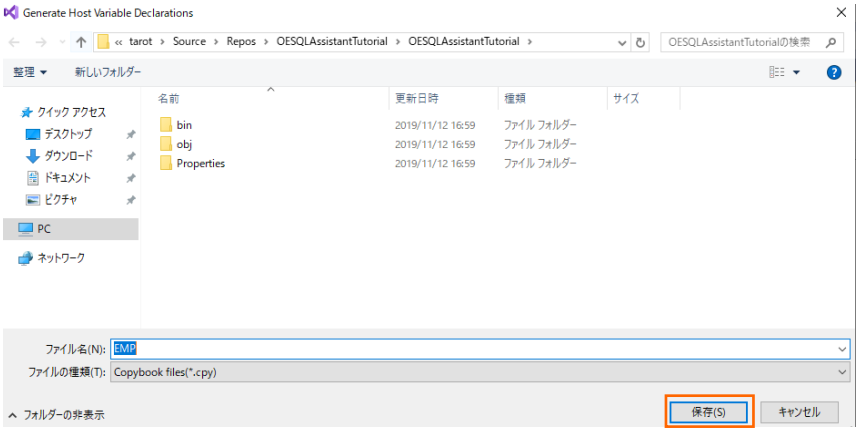
```
*> Put your program logic/SQL statements here
     EXEC SQL
     DECLARE CSR4 CURSOR FOR SELECT
             A.FIRSTNME
            ,A.LASTNAME
            ,A.SALARY
       FROM dbo.EMP A
    END-EXEC
    EXEC SQL
       OPEN CSR4
    END-EXEC
    PERFORM UNTIL SQLCODE < 0 OR SQLCODE = +100
       EXEC SQL
         FETCH CSR4  INTO
             :EMP-FIRSTNME
            ,:EMP-LASTNAME
            ,:EMP-SALARY:EMP-SALARY-NULL
       END-EXEC
       *> Process data from the Fetch
       IF SQLCODE = 0

           DISPLAY 'ROW FOUND'
```

2) ホスト変数項目の追加

① OpenESQL アシスタントを表示し、「EMP(dbo)」を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[DCLGEN を生成する(D)] を選択します。



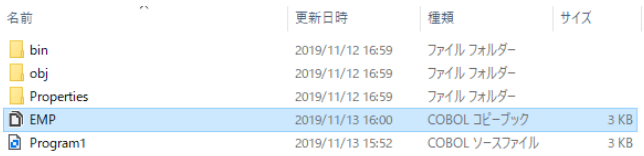
② そのまま、[保存(S)] をクリックします。

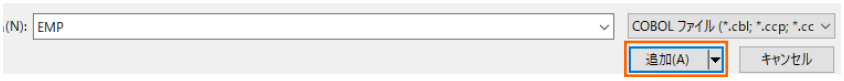


補足）
デフォルトのファイル名は、テーブル名が表示されます。
デフォルトの保存先は、開いているプロジェクト（今回は、OESQLAssistantTutorial）の直下です。

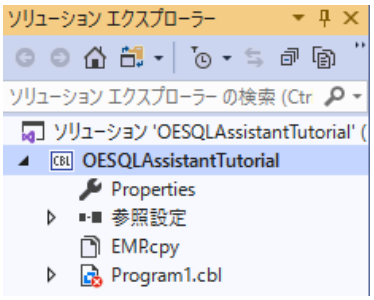
③ ソリューションエクスプローラーより、OpenESQLAssistantTutorial プロジェクトを選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[追加(D)] > [既存の項目(G)] を選択します。

④ 「EMP」COBOL コピーブックを選択し、[追加(A)] をクリックします。





EMP.cpy がプロジェクト配下に追加されます。

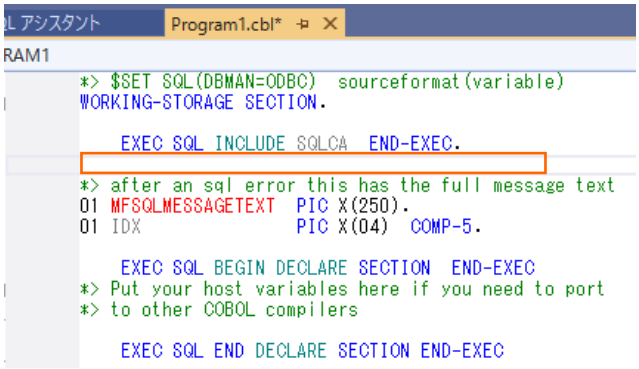


EMP.cpy をダブルクリックすると、以下のような COBOL プログラムが表示されます。

```
*> ------------------------------------------
*> DECLARE TABLE for EMP
*> ------------------------------------------
    EXEC SQL DECLARE EMP TABLE
    ( EMPNO                 char(6)      NOT NULL
    , FIRSTNME              varchar(12)  NOT NULL
    , MIDINIT               char(1)      NOT NULL
    , LASTNAME              varchar(15)  NOT NULL
    , WORKDEPT              char(3)
    , PHONENO               char(4)
    , HIREDATE              date
    , JOB                   char(8)
    , EDLEVEL               smallint
    , SEX                   char(1)
    , BIRTHDATE             date
    , SALARY                decimal(9,2)
    , BONUS                 decimal(9,2)
    , COMM                  decimal(9,2)
    ) END-EXEC.
*> ------------------------------------------
*> COBOL HOST VARIABLES FOR TABLE EMP
*> ------------------------------------------
 01  EMP-EMPNO                          STRING.
 01  EMP-FIRSTNME                       STRING.
 01  EMP-MIDINIT                        STRING.
 01  EMP-LASTNAME                       STRING.
 01  EMP-WORKDEPT                       STRING.
 01  EMP-PHONENO                        STRING.
 01  EMP-HIREDATE                       type System.DateTime.
 01  EMP-JOB                            STRING.
 01  EMP-EDLEVEL                        BINARY-SHORT.
 01  EMP-SEX                            STRING.
 01  EMP-BIRTHDATE                      type System.DateTime.
 01  EMP-SALARY                         type System.Double.
 01  EMP-BONUS                          type System.Double.
 01  EMP-COMM                           type System.Double.
*> ------------------------------------------
*> COBOL INDICATOR VARIABLES FOR TABLE EMP
*> ------------------------------------------
```

OpenESQL アシスタントが、選択したテーブル構造を基にホスト変数など、必要な定義が記述されています。

⑤ 上記コピー句を Program1.cbl の適切な位置に挿入するため、ソリューションエクスプローラーより「Proram1.cbl」をダブルクリックします。そのうえで、”EXEC SQL INCLUDE SQLCA END-EXEC.” 句の次の空行をクリックします。



⑥ OpenESQL アシスタントを開き、「補助コード」タブを選択すると、EXEC SQL INCLUDE 句が指定されています。「現在のクエリーにプログラムを挿入する」アイコンをクリックします。

再度、Program1.cbl に戻ると、EMP コピーブックの INCLUDE が挿入されています。

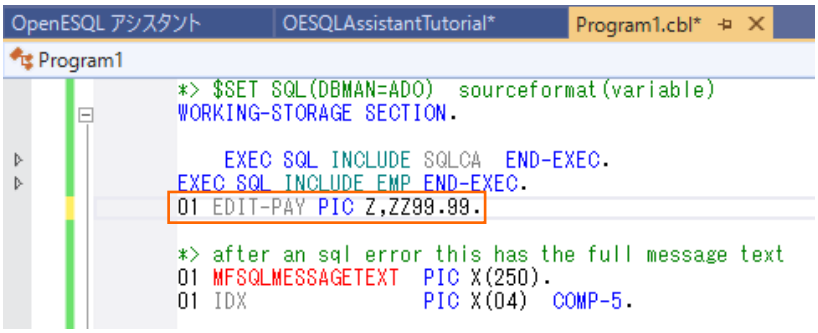
```
*> $SET SQL(DBMAN=ODBC)  sourceformat(variable)
WORKING-STORAGE SECTION.

    EXEC SQL INCLUDE SQLCA  END-EXEC.
EXEC SQL INCLUDE EMP END-EXEC.
```

⑦ Program1.cbl の変更を保存します。

3) 実行時のコンソールログの追加

① Program1.cbl を開き、” EXEC SQL INCLUDE EMP END-EXEC” の後に以下の行を追加します。

“01 EDIT-PAY PIC Z,ZZ99.99.”



② “DISPLAY 'ROW FOUND'” 行を以下の２行と置換します。

“MOVE EMP-SALARY TO EDIT-PAY”

“DISPLAY 'NAME: ' EMP-LASTNAME ', ' EMP-FIRSTNME ' SALARY: ' EDIT-PAY UPON CONSOLE”

変更前

```
*> Process data from the Fetch
IF SQLCODE = 0

    DISPLAY 'ROW FOUND'

*> for array fetches, field sqlerrd(3) contains
*> the number of rows returned
*>   PERFORM VARYING IDX FROM 1 BY 1
*>     UNTIL IDX > SQLERRD(3)

*>   you will need to add code here to process the array

*>   END-PERFORM
END-IF
```

変更後

```
      *> Process data from the Fetch
      IF SQLCODE = 0

           MOVE EMP-SALARY TO EDIT-PAY
           DISPLAY 'NAME: ' EMP-LASTNAME ', ' EMP-FIRSTNME
              '   SALARY: ' EDIT-PAY UPON CONSOLE

      *> for array fetches, field sqlerrd(3) contains
      *> the number of rows returned
      *>   PERFORM VARYING IDX FROM 1 BY 1
      *>     UNTIL IDX > SQLERRD(3)

      *>   you will need to add code here to process the array

      *>   END-PERFORM
      END-IF
```

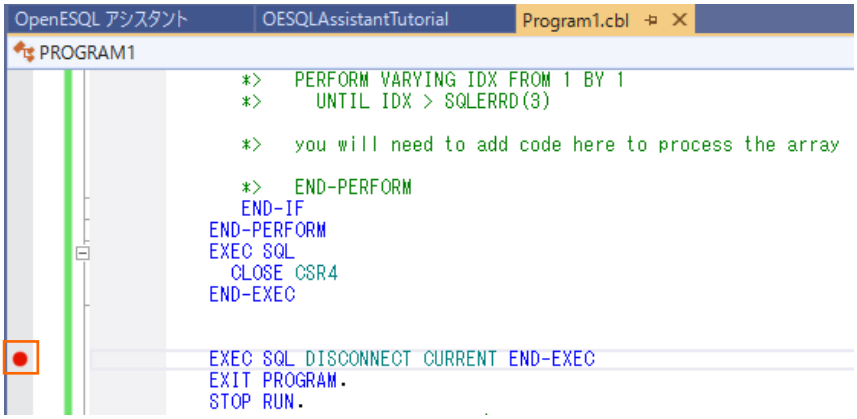
③ 変更を保存します。

3.6. SQL 埋め込みプログラムの実行

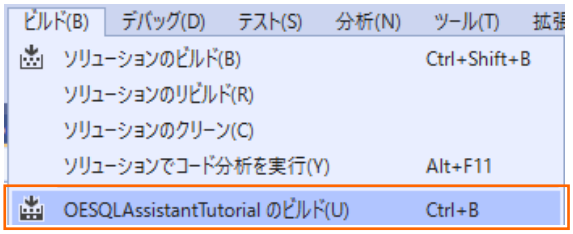
1) Program1.cbl を開き、” EXEC SQL DISCONNECT CURRENT END-EXEC” の行をクリックし、IDE のメニューよ

り [デバッグ(D)] > [ブレークポイントの設定/解除(G)] を選択し、ブレークポイントを設定します。

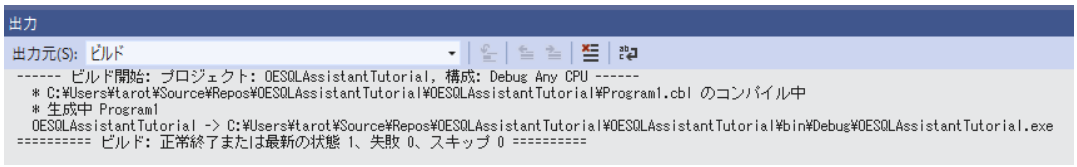
ブレークポイントが設定された箇所には、赤丸が設定されます。



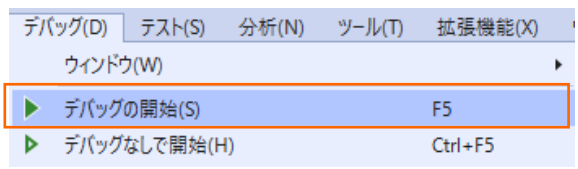
2) IDE のメニューより [ビルド(B)] > [OESQLAssistantTutorial のビルド(U)] をクリックします。



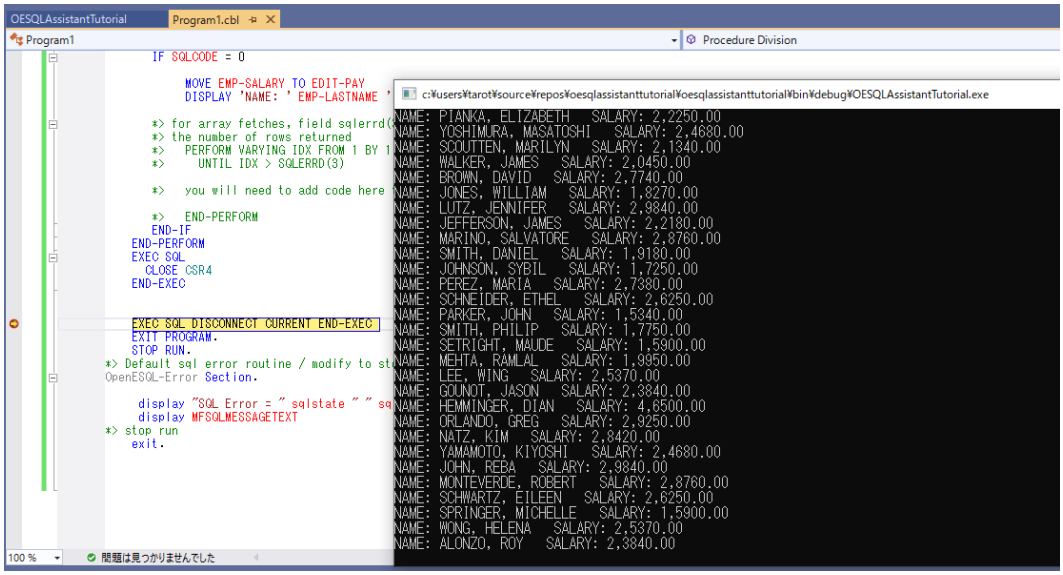
ビルドが正常に終了します。



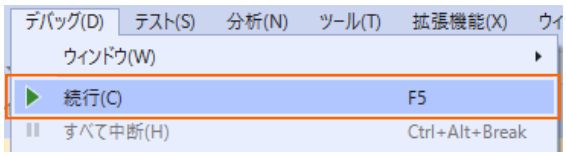
3) IDE のメニューより [デバッグ(D)] > [デバッグの開始(S)] をクリックして、デバッグ起動します。



ブレークポイントで停止し、コンソール画面に検索結果が一覧で表示されます。



IDE のメニューより [デバッグ(D)] > [続行(C)] をクリックするとデバッグが終了します。



WHAT'S NEXT

- 本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。

免責事項

ここで紹介したソースコードは、機能説明のためのサンプルであり、製品の一部ではございません。ソースコードが実際に動作するか、御社業務に適合するかなどに関しまして、一切の保証はございません。 ソースコード、説明、その他すべてについて、無謬性は保障されません。

ここで紹介するソースコードの一部、もしくは全部について、弊社に断りなく、御社の内部に組み込み、そのままご利用頂いても構いません。

本ソースコードの一部もしくは全部を二次的著作物に対して引用する場合、著作権法の精神に基づき、適切な扱いを行ってください。