Windows における COBOL とデータベースの連携

東京システムハウス株式会社
発行：2015 年 3 月 31 日

目次

1　はじめに

1.1　COBOL とデータベースの連携

1960 年に誕生した COBOL は現在でも多くの企業や組織の基幹業務システムとして利用され社会の基盤を支えている。これらのシステムは従来オフコンや汎用機の上で稼働していたが、運用保守費用のコスト削減や COBOL 技術者の後継者不足、また、サポート終了などの理由によりオフコンからオープン環境への資産移行が行われつつある。

オープン化の際、データベースについては従来の階層型(RDB)やネットワーク型(NDB)から、現在主流とされているリレーショナル型(RDBMS)への移行が課題となる。

本ホワイトペーパーでは、オープン化の一例としてオープンソースとして公開されている opensource COBOL に焦点を置き、Windows 上での opensource COBOL と RDBMS の一つである PostgreSQL との連携について記載する。

1.2　opensource COBOL とは

opensource COBOL とは、日本特有のビジネス環境に即したカスタマイズやメンテナンスを加えた、日本発のオープンソース COBOL コンパイラである。開発の主体は OSS コンソーシアムのオープン COBOL ソリューション部会であり、2012 年に OpenCOBOL 1.1 Pre-release よりフォークされている。
※OpenCOBOL は、日本医師会総合政策研究機構 ORCA プロジェクトで開発が始められ、その原作者は西田圭介氏である

また opensource COBOL に対応したデータベース連携ツールとして、埋め込み SQL に対応した Open Cobol ESQL ならびにファイル I/O 命令に対応した Open Cobol DB Library の開発も行っている。従来は Linux のみに対応していたが、最新版では Windows もサポートしている。連携先のデータベースとして現在は PostgreSQL をサポートしている。

前述の 2 つの連携ツールのうち Open Cobol ESQL は OSS コンソーシアムのオープン COBOL ソリューション部会にて公開されている。

1.3　埋め込み SQL

1.3.1 概要

埋め込み SQL は、ホスト変数を利用して COBOL プログラムとデータベースの連携を実現する。データベースに対する操作は COBOL プログラム中に SQL を直接記述することで行うことができる。

リスト①は、COBOL プログラムとデータベース相互間でのデータの受け渡しに利用するホスト変数の定義と、SQL を実行した時のエラーコードやエラーメッセージを格納する共通領域(SQLCA)の定義を行っている。

[リスト①]

```
DATA DIVISION.
WORKING-STORAGE SECTION.
EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC SQL BEGIN DECLARE SECTION END-EXEC..
01 HVCNT        PIC 9(5).
01 HVBCD        PIC 9(3).
EXEC SQL END DECLARE SECTION END-EXEC.
```

リスト②は、ホスト変数を利用した埋め込み SQL の例で、記述例①で定義したホスト変数が SQL で利用されている。このとき、ホスト変数の先頭にはコロン(:)を付ける必要がある。SQL の実行結果(成功／失敗)については SQLCA に格納されるエラーコードを確認することで判断できる。

[リスト②]

```
 [PROCEDURE DEVISION]
(中略)
MOVE 1 TO HVBCD.
EXEC SQL
  SELECT COUNT(ID) INTO :HVCNT
  FROM M_SYAIN WHERE BUSYOCD = :HVBCD
END-EXEC.
DISPLAY “COUNT: “ HVCNT.
```

1.3.2 Open Cobol ESQL によるデータベース連携

Open Cobol ESQL は、埋め込み SQL に対応したデータベース連携ツールである。Open Cobol ESQL はプリコンパイラ(ocesql)とライブラリ(libocesql)の総称で、COBOL プログラム内の埋め込み SQL をプリコンパイラによりライブラリ API を call で呼び出す形に変換することで COBOL プログラムとデータベースの連携を実現している。

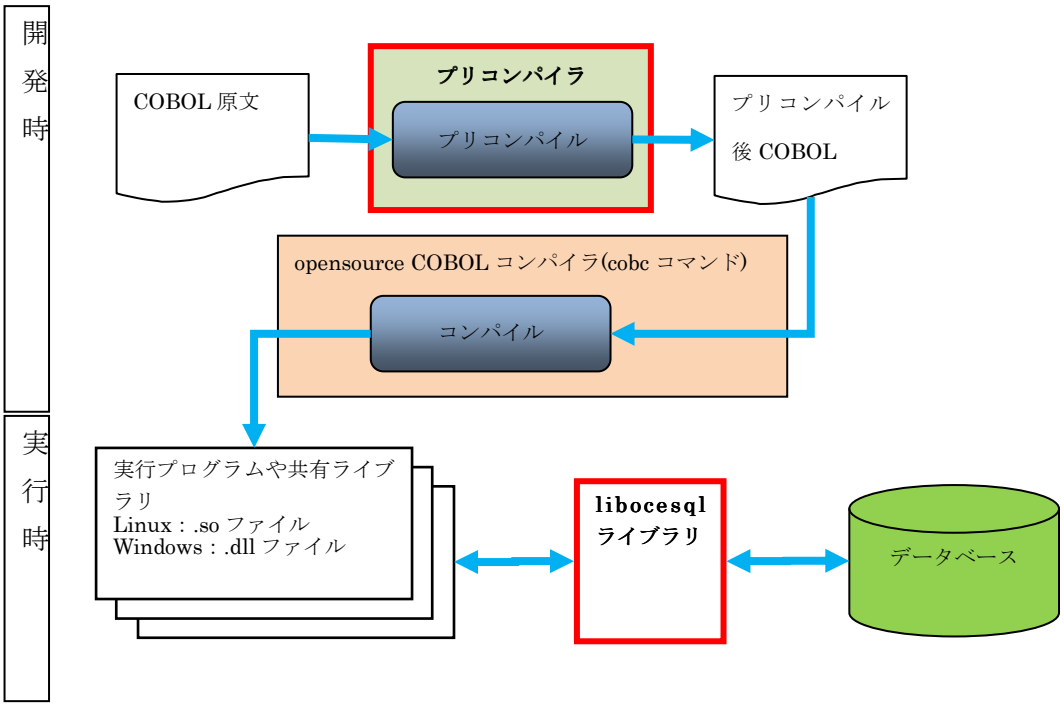


図 1.3.2-1. Open Cobol ESQL によるデータベース連携イメージ

1.4 ファイル I/O 命令

1.4.1 概要

通常のファイル操作と同じ要領でファイル I/O 命令(READ, WRITE)によるデータベースへのデータの書き込み、読み込みをサポートする COBOL 製品もある。ファイル I/O 命令がそのままデータベースに対する操作となるため、データをファイル管理からデータベース管理に移行する場合 COBOL プログラムの改修が少なり、SQL の経験が浅い COBOL 技術者でも開発を行いやすいという利点がある。

1.4.2 Open Cobol DB Library によるデータベース連携

Open Cobol DB Library は、ファイル I/O 命令に対応したデータベース連携ツー

ルである。opensource COBOL はファイル I/O 命令実行時に環境変数で設定したファイルハンドラに制御を渡すことができる。制御が渡ったファイルハンドラ内でライブラリ(libocdblibrary)の API を呼び出すことで、COBOL のファイル I/O 命令によるデータベース連携を実現している。

ファイル I/O 命令の場合 1 つのファイルがデータベース上の 1 つのテーブルに対応しており、Open Cobol DB Library ではファイル単位で環境変数を設定することで、一部のファイルのみデータベースと連携することが可能となる。この事から、COBOL 開発者はプログラム上でファイルとデータベースの区別を意識せずに処理を記述する事ができる。

Open Cobol DB Library は、データベースと連携する際にファイル定義ファイルを読み込む必要がある。ファイル定義ファイルは、COBOL プログラムの FD 項目で定義されているフィールド名やサイズなどの情報を記述したもので、Open Cobol DB Library の付属ツール(図 1.4.2-1 の JDDGenerator)から生成する事ができる。

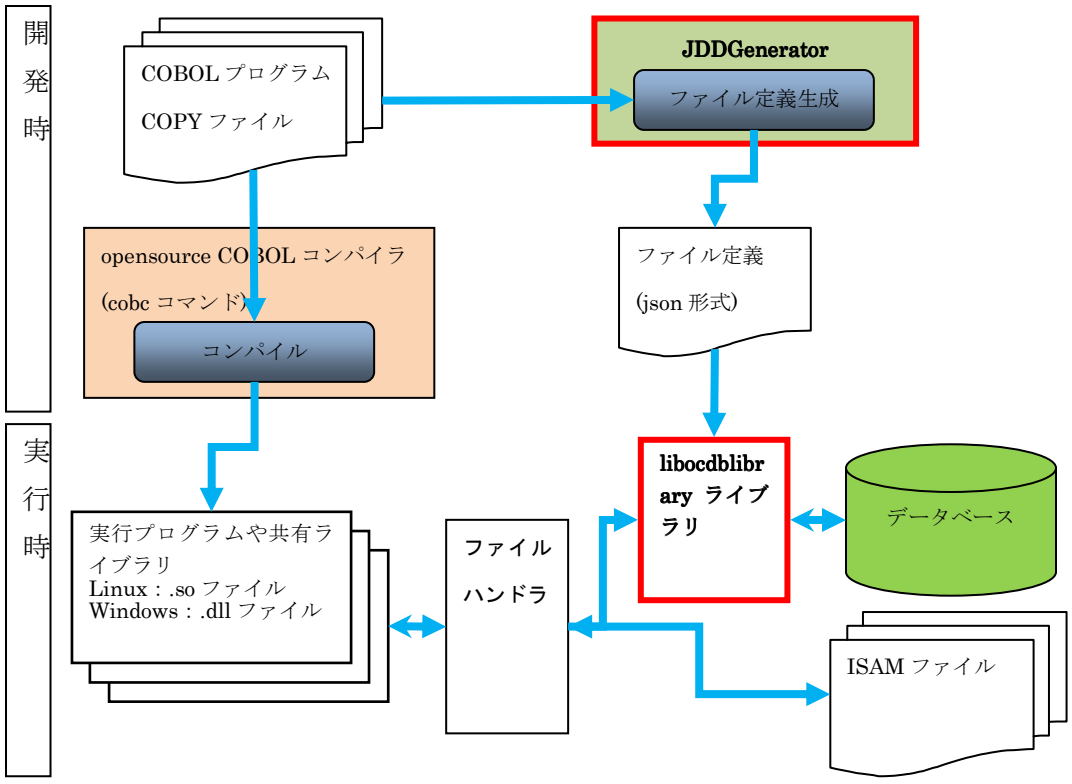


図 1.4.2-2. Open Cobol DB Library によるデータベース連携イメージ

2　動作環境

opensource COBOL とデータベース連携の確認は、以下の環境で行った。

- 環境
  Microsoft Windows Server 2012 R2 Standard
  Microsoft Visual Studio Professional 2013
  Microsoft Windows7 Professional Service Pack 1

- 関連ツール
  PostgreSQL 9.3.6 (x86-32)
  opensource COBOL 1.4.0J

3　インストール

3.1　入手先について

本ホワイトペーパーで利用するツールのうち opensource COBOL、OCESQ、PostgreSQL はダウンロードサイトが用意されている。

3.1.1　opensource COBOL および Open Cobol ESQL

opensource COBOL および Open Cobol ESQL は、OSS コンソーシアム様のサイトでオープンソースソフトウェアとして公開している。

- OSS コンソーシアム
  http://www.osscons.jp/
- ダウンロードページ
  - opensource COBOL v1.4.0J
    http://www.osscons.jp/osscobol/download/v1_4j/
    ファイル名：opensourceCOBOL-1.4.0J.zip

  - Open Cobol ESQL v1.1.0
    http://www.osscons.jp/osscobol/download/addtools/
    ファイル名：ocesql-1.1.0.tar.gz

3.1.2 Open Cobol DB Library

Open Cobol DB Library は東京システムハウス株式会社より提供を行っている。

3.1.3 PostgreSQL

Windows 版 PostgreSQL のインストーラーは、EnterpriseDB 社のサイトからダウンロードできる。

- EnterpriseDB 社
  http://www.enterprisedb.com/
- ダウンロードページ
  http://www.enterprisedb.com/products-services-training/pgdownload
  ファイル名：Installer version 9.3.6 Win x86-32

3.2 opensource COBOL のインストール

opensource COBOL のインストール方法については、opensource COBOL パッケージに含まれている README を参照されたい。今回のインストール環境は以下の通りである。

OS	Windows Server 2012 R2
開発環境	Visual Studio 2013 professional
インストール先	D:¥develop¥work¥oscobol
対象プラットフォーム	Win32
対象コンフィグレーション	Release

表 3.1.3-1. opensource COBOL のインストール環境

3.3 PostgreSQL のインストール

3.1.3 節のダウンロードサイトでダウンロードしたアプリケーション形式のインストーラからインストールを行うことができる。インストーラはウィザード形式になっており、今回の動作環境では表 3.3.1-1 に記載した情報を入力してインストールを実行した。

OS	Windows Server 2012 R2
インストール先	C:¥Program Files (x86)¥PostgreSQL¥9.3
データ保存先	C:¥Program Files (x86)¥PostgreSQL¥9.3¥data
スーパーユーザーのパスワード	postgres
サーバの待ち受けポート	5432
ロケール	Japanese, Japan

表 3.3-1 PostgreSQL のインストール設定例

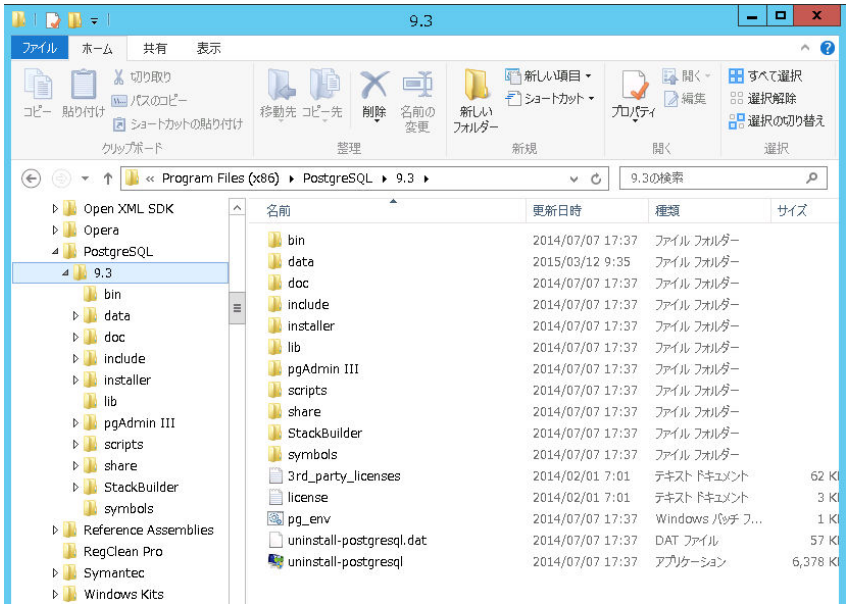


図 3.3-2. インストールされた PostgreSQL フォルダ

3.4 Open Cobol ESQL のインストール

3.4.1 インストール環境

今回のインストール環境は以下の通りである。

OS	Windows Server 2012 R2
開発環境	Visual Studio 2013 professional
対象プラットフォーム	Win32
対象コンフィグレーション	Release

表 3.4.1-1. Open Cobol ESQL のインストール環境

3.4.2 インストール手順

Open Cobol ESQL パッケージ(ocesql-1.1.0.tar.gz) を任意のフォルダで解凍する。解凍後、win32 フォルダにあるソリューションファイル ocesql.sln を Microsoft Visual Studio で開く。

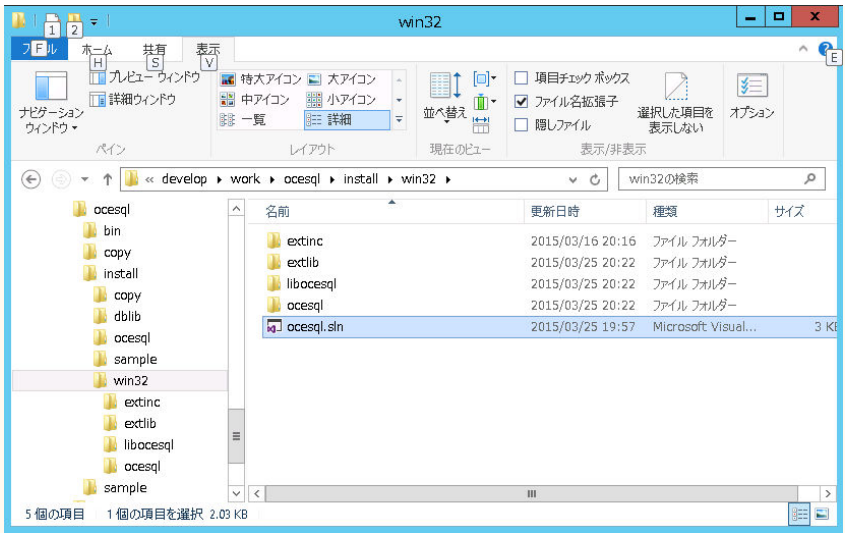


図 3.4.2-1. ocesql.sln ファイル

ソリューションファイルの中には libocesql と ocesql の 2 つのプロジェクトがあり、ツールバーから「ビルド（B)」→「ソリューションのビルド（B)」の操作、もしくは、ソリューションエクスプローラーから「ソリューションのビルド」を選択し、ビルドを実行する。

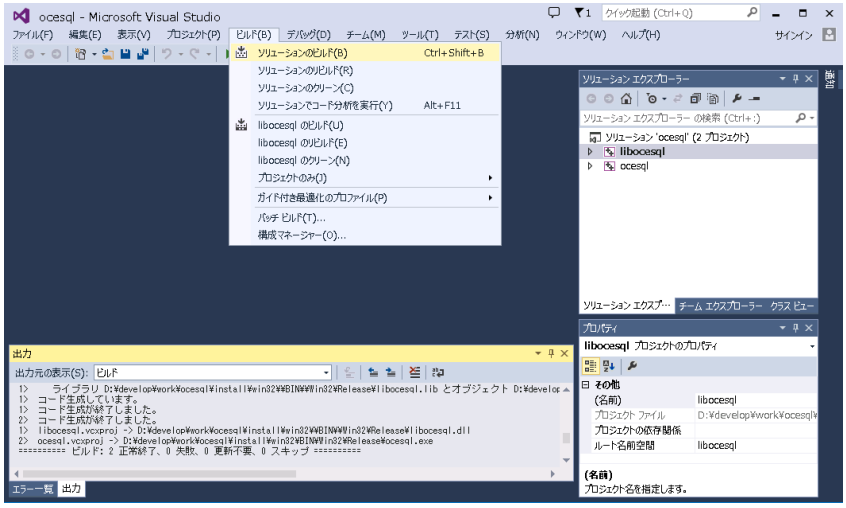


図 3.4.2-2. ocesql.sln のビルド実行

ビルドに成功すると、BIN¥Win32¥Release フォルダの中に Open Cobol ESQL ライブラリ(libocesql.dll, libocesql.lib)及びプリコンパイラ(ocesql.exe)が生成される。

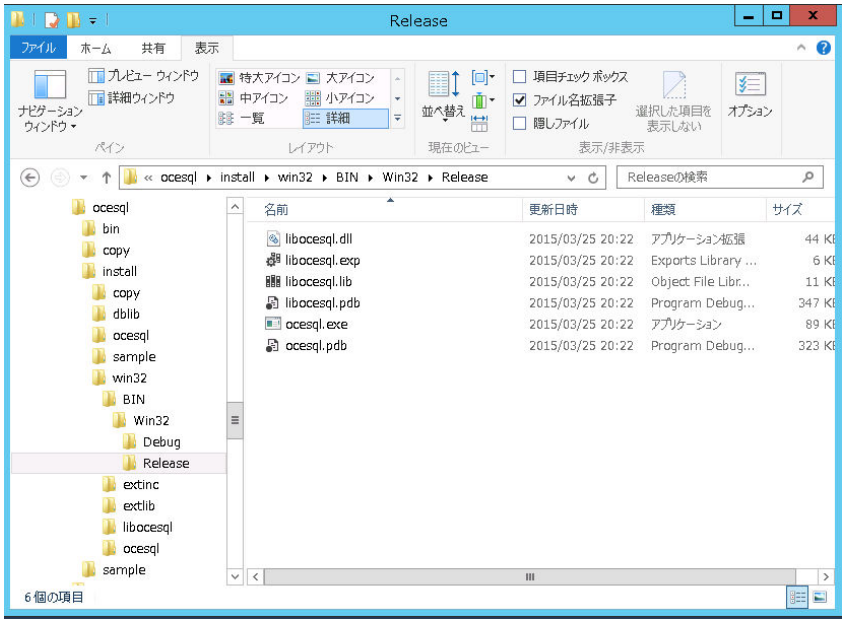


図 3.4.2-3. BIN¥Win32¥Release フォルダに出力される内容

opensource COBOL の cobc や cobcrun の実行時に Open Cobol ESQL ライブラリを参照できるように、libocesq.dll と libocesq.lib を opensource COBOL の環境変数 LIB で設定したディレクトリに配置する。

3.5 Open Cobol DB Library のインストール

3.5.1 インストール環境

今回のインストール環境は以下の通りである。

OS	Windows Server 2012 R2
開発環境	Visual Studio 2013 professional
対象プラットフォーム	Win32
対象コンフィグレーション	Release

表 3.5.1-1. Open Cobol DB Library のインストール環境

3.5.1 インストール手順

Open Cobol DB Library パッケージを任意のフォルダで解凍する。解凍後、win32 フォルダにあるソリューションファイル ocdblibrary.sln を Microsoft Visual

Studio で開く。

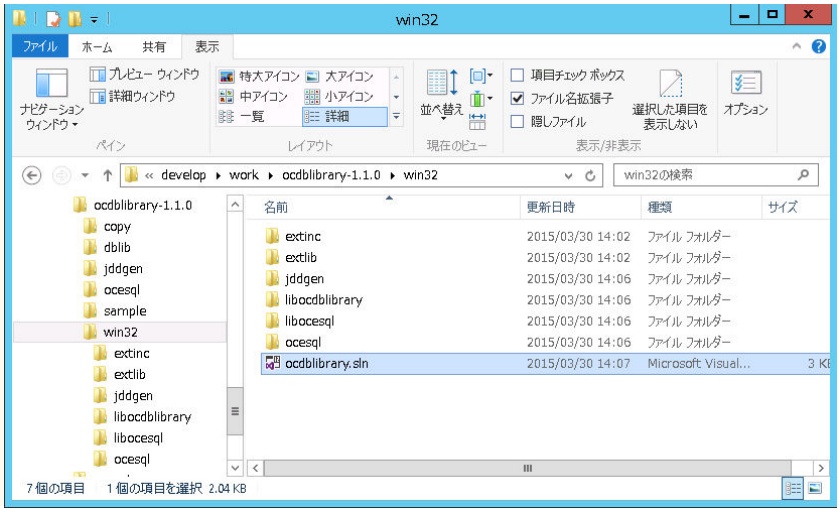


図 3.5.1-1. ocdblibrary.sln ファイル

ソリューションの中には libocdblibrary と jddgen の 2 つのプロジェクトがあり、ツールバーから「ビルド（B)」→「ソリューションのビルド（B)」の操作、もしくは、ソリューションエクスプローラーから「ソリューションのビルド」を選択し、ビルドを実行する。

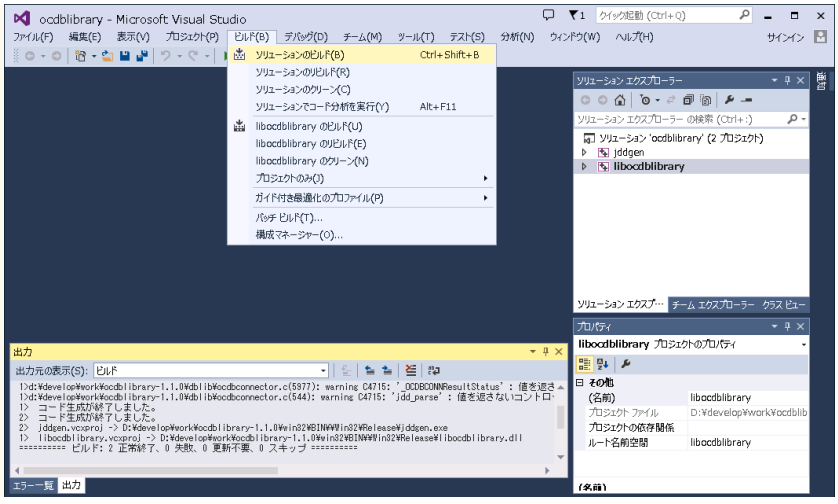


図 3.5.1-2. ocdblibrary.sln のビルド実行

ビルドに成功すると、BIN¥Win32¥Release フォルダの中に Open Cobol DB Library ライブラリ(libocdblibrary.dll, libocdblibrary.lib)及びファイル定義ファイル生成ツール(jddgenerator.exe)が生成される。

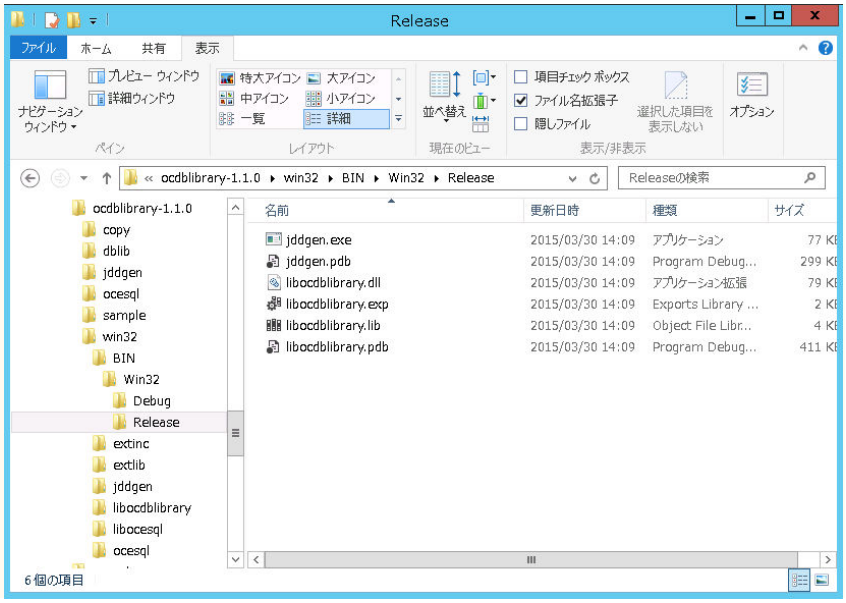


図 3.5.1-3. BIN¥Win32¥Release フォルダに出力される内容

opensource COBOL の cobc や cobcrun の実行時に Open Cobol DB Library ライブラリを参照できるように、libocdblibrary.dll と libocdblibrary.lib を opensource COBOL の環境変数 LIB で設定したディレクトリに配置する。

4 埋め込み SQL のコンパイル及び実行

本章では、パッケージに同梱しているサンプルプログラムを用い、データベースの準備、コンパイル、コンパイル後の実行までの操作手順を例を用い説明する。

4.1 データベースの準備

PostgreSQL に、pgAdmin III を使用して COBOL と連携するデータベースを作成する。

pgAdmin III を起動し、予め作成されているサーバ PostgreSQL9.3(x86)に新規データベース testdb を作成する。オブジェクトブラウザの PostgreSQL9.3(x86)を右クリックし「新しいデータベース」を選択する。

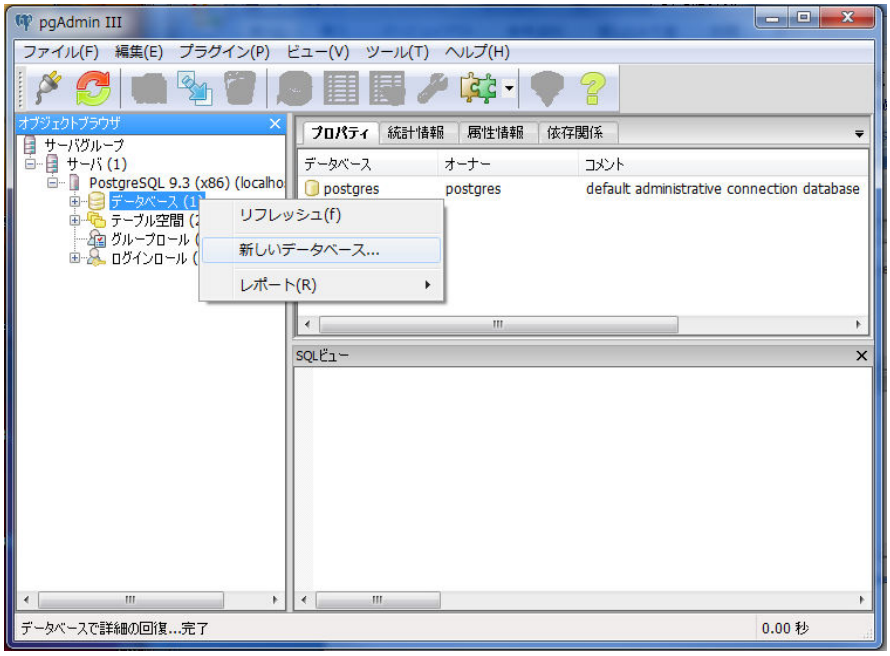


図 4.1-1.pgAdmin III の画面

「新しいデータベース」ウィンドウ選択後表示されるダイアログのプロパティタブに、下記の情報を設定した後、「OK」ボタンを押す

- 名前：testdb
- オーナー：postgres

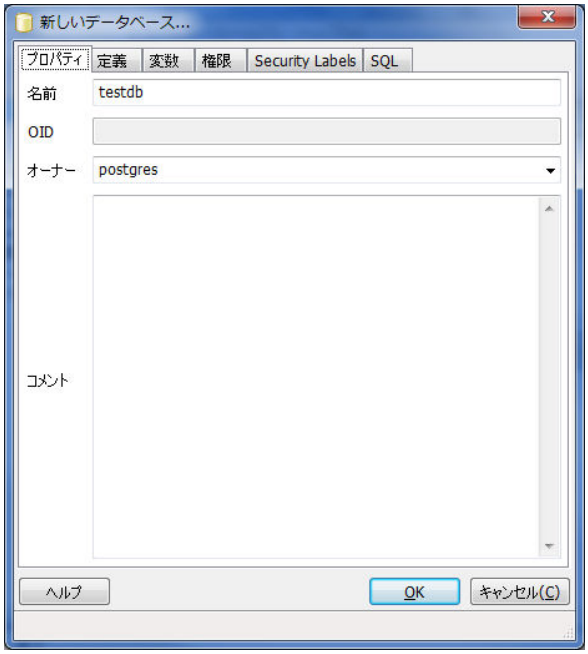


図 4.1-2. データベース作成ダイアログ

pgAdmin III のオブジェクトブラウザ／データベースに”testdb”が追加されている事を確認する。

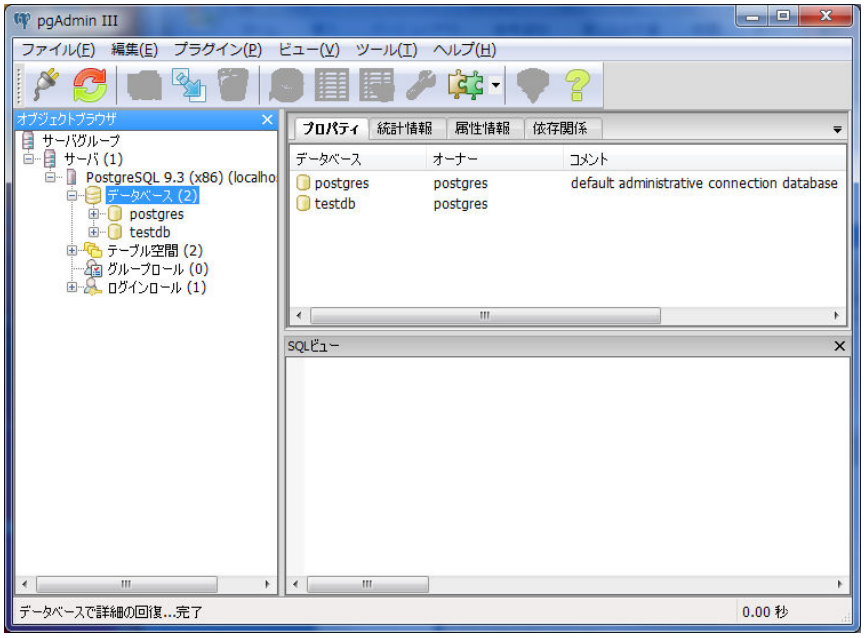


図 4.1-3. データベース作成後の pgAdmin III 画面

4.2 COBOL ソースファイルのコンパイル

4.2.1 準備

本節の作業は任意のワーキングフォルダで行う。

- Open Cobol ESQL パッケージ

3.3 節で利用した Open Cobol ESQL パッケージから、以下のファイルをワーキングフォルダにコピーする。

  - sample 及び copy フォルダ
    フォルダごとコピーする
  - BIN フォルダ内に生成された ocesql. exe
    ワーキングフォルダに bin フォルダを作成し、その中にコピーする

以降の作業はコマンドプロンプトから行う。

- CL コンパイラの指定

Windows 版の opensource COBOL は Microsoft の CL コンパイラを利用する。CL コンパイラは CPU ごとに異なっており、どのコンパイラを利用するかコマンドラインから vcvarsall を実行して指定する必要がある。詳細は下記 MSDN サイ

トを参照すること。

https://msdn.microsoft.com/ja-jp/library/f2ccy3wt.aspx

図 4.2.1-1.vcvarsall.exe 実行

- 環境変数の設定

PostgreSQL インタフェースである libpq ライブラリや 3.4 節で生成した libocesql ライブラリを利用するには環境変数の設定が必要となる。設定対象となる環境変数と、今回設定した値の表は下記の通りである。

PATH	%WORK%¥bin;"C:¥Program Files (x86)¥PostgreSQL¥9.3¥bin";%PATH% (※)環境変数 WORK はワーキングフォルダの場所を意味する
COB_PRE_LOAD	libocesql

表 4.2.1-1. 環境変数の設定内容

4.2.2 プリコンパイル及びコンパイル

環境変数の設定完了後、ワーキングフォルダの COBOL を ocesql コマンドでプリコンパイルする。

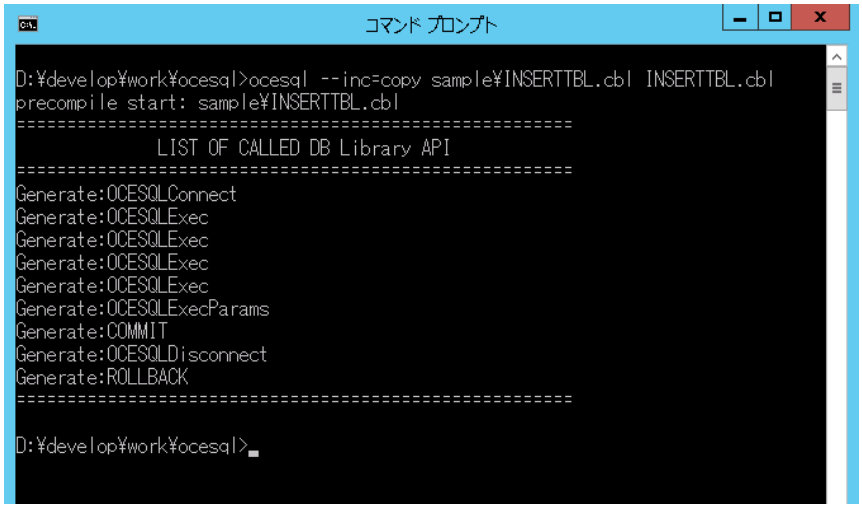


図 4.2.2-1. プリコンパイル実行

プリコンパイル後のファイルに対し cobc コマンドでコンパイルを行い、DLL ファイルを生成する。

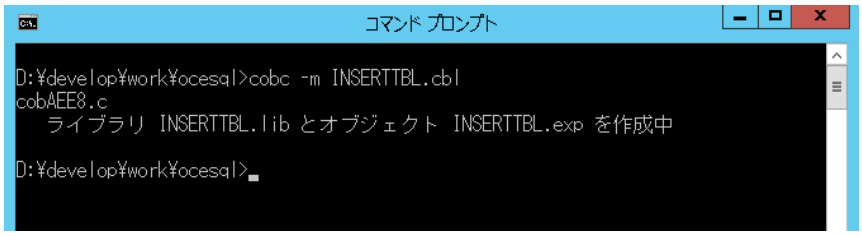


図 4.2.2-2. コンパイル実行

4.3 実行

コンパイル後に出来た INSERTTBL.dll, FETCHTBL.dll は cobcrun コマンドを用いて実行する。以下のサンプルプログラム実行画面は、INSERTTBL の実行でテーブル EMP の作成とレコードの挿入が行われ、FETCHTBL の実行により挿入したレコードの取得及び表示が正常に行われている事を表している。

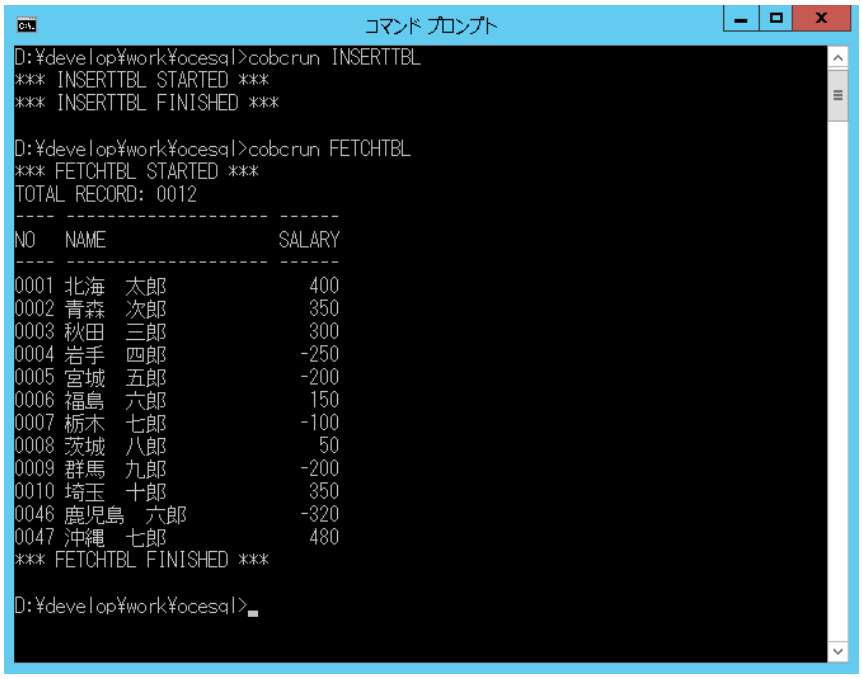


図 4.3-1. サンプルプログラムの実行

4.4 既知の制約事項と問題点

既知の制約事項及び問題点は以下の通り。これらは、今後の機能拡張時の課題とする。

- COMP-X など、一部の型に未対応

- EXEC SQL WHENEVER 文に未対応
- プリコンパイラの COPY 句及びフリーフォーマットに未対応
- 接続できるデータベースは PostgreSQL のみ
- カラム名に大文字を指定した時、小文字で PostgreSQL に登録される

5 ファイル I/O 命令の実行

本章では Open Cobol DB Library を用いたファイル I/O 命令によるデータベース連携の実行例を説明する。連携するデータベースは 4 章で作成したデータベース testdb を利用する。

5.1 実行する COBOL ファイルの概要

ファイルから読み込んだデータをデータベースに格納するプログラム(INSERTTBL.cbl)と、前者のプログラムで格納したデータをフェッチしてディスプレイに表示するプログラム(FETCHTBL.cbl)の 2 本について解説する。

- データベースに格納するプログラム(INSERTTBL.cbl)

```
       IDENTIFICATION              DIVISION.
       PROGRAM-ID.                 INSERTTBL.
       ENVIRONMENT                 DIVISION.
       INPUT-OUTPUT                SECTION.
       FILE-CONTROL.
       SELECT DB-FILE
           ASSIGN EXTERNAL IOTABLE
           ACCESS SEQUENTIAL
           ORGANIZATION IS INDEXED
           RECORD KEY IS DR_NO
           FILE STATUS IS FTEST-STATUS.
       SELECT FTEST-FILE
           ASSIGN EXTERNAL "test.data"
           ORGANIZATION LINE SEQUENTIAL
           FILE STATUS IS FTEST-STATUS.
      *
       DATA                        DIVISION.
       FILE                        SECTION.
       FD  DB-FILE.
       01  DB-RECORD.
           05  DR_NO               PIC  9(2).
           05  DR_LANK             PIC  9(2).
           05  DR_TEAM             PIC  N(10).
           05  DR_POINT            PIC  9(4).
           05  DR_GOAL_DIF         PIC  S9(4).
       FD  FTEST-FILE.
       01  FTEST-RECORD.
           05  FR_NO               PIC  9(2).
           05  FR_LANK             PIC  9(2).
           05  FR_TEAM             PIC  N(10).
           05  FR_POINT            PIC  9(4).
           05  FR_GOAL_DIF         PIC  S9(4).
       WORKING-STORAGE              SECTION.
       77  FTEST-STATUS            PIC  X(2) VALUE "00".
```

図 5.1-1. INSERTTBL プログラム(1/2)

```
       PROCEDURE                   DIVISION.
       MAIN                        SECTION.
       MAIN-000.
           OPEN INPUT FTEST-FILE.
           OPEN OUTPUT DB-FILE.
           PERFORM UNTIL FTEST-STATUS NOT = "00"
               READ FTEST-FILE NEXT RECORD
                   END CONTINUE
                   NOT END
                   MOVE  FR_NO TO DR_NO
                   MOVE  FR_LANK TO DR_LANK
                   MOVE  FR_TEAM TO DR_TEAM
                   MOVE  FR_POINT TO DR_POINT
                   MOVE  FR_GOAL_DIF TO DR_GOAL_DIF
                   WRITE DB-RECORD
               END-READ
           END-PERFORM.
           CLOSE FTEST-FILE.
           CLOSE DB-FILE.
       MAIN-999.
           STOP RUN.
```

図 5.1-2. INSERTTBL プログラム(2/2)

INSERTTBL 内で参照するデータファイル test.data の内容は以下の通り。

```
0101 東京イーグルス       00640057
0205 大阪ブレーブス       00390006
0309 クリスタル仙台       0034001r
0413 横浜アスレチックス   0029001r
0517 広島ユナイテッド     0024002v
```

図 4.2.2-3. INSERTTBL プログラム用サンプルデータ

- データをフェッチして表示するプログラム(FETCHTBL.cbl)

```
       IDENTIFICATION              DIVISION.
       PROGRAM-ID.                 FETCHTBL.
      *
       ENVIRONMENT                 DIVISION.
       INPUT-OUTPUT                SECTION.
       FILE-CONTROL.
       SELECT DB-FILE
           ASSIGN EXTERNAL IOTABLE
           ACCESS SEQUENTIAL
           ORGANIZATION IS INDEXED
           RECORD KEY IS DR_NO
           FILE STATUS IS FTEST-STATUS.
      *
       DATA                        DIVISION.
       FILE                        SECTION.
       FD  DB-FILE.
       01  DB-RECORD.
            05  DR_NO              PIC  9(2).
            05  DR_LANK            PIC  9(2).
            05  DR_TEAM            PIC  N(10).
            05  DR_POINT           PIC  9(4).
            05  DR_GOAL_DIF        PIC  S9(4).
       WORKING-STORAGE             SECTION.
       01  D-HEAD.
            05  FILLER             PIC  N(1) VALUE "順".
            05  FILLER             PIC  X VALUE "|".
            05  FILLER             PIC  N(10) VALUE "チーム".
            05  FILLER             PIC  X VALUE "|".
            05  FILLER             PIC  N(2) VALUE "勝点".
            05  FILLER             PIC  X VALUE "|".
            05  FILLER             PIC  N(4) VALUE "得失差".
```

図 5.1-4. FETCHTBL プログラム(1/2)

```
01  D-REC.
     05  D-LANK              PIC  Z9.
     05  FILLER              PIC  X  VALUE "|".
     05  D-TEAM              PIC  N(10).
     05  FILLER              PIC  X VALUE "|".
     05  D-POINT             PIC  Z(3)9.
     05  FILLER              PIC  X  VALUE "|".
     05  D-GOAL-DIF          PIC  --,--9.
77  FTEST-STATUS            PIC  X(2) VALUE "00".
PROCEDURE                    DIVISION.
MAIN                         SECTION.
MAIN-000.
    OPEN INPUT DB-FILE.
    DISPLAY D-HEAD
    PERFORM UNTIL FTEST-STATUS NOT = "00"
        READ DB-FILE NEXT RECORD
            END CONTINUE
            NOT END
            MOVE  DR_LANK TO D-LANK
```

図 5.1-5. FETCHTBL プログラム(2/2)

5.2 ファイル定義ファイルの生成

ファイル定義ファイルを生成する JDDGenerator は Open Cobol DB Library に同梱されており、Open Cobol DB Library プロジェクトをビルドすることで jddgen.exe というアプリケーション名で出力され利用できるようになる。

今回のサンプルプログラムの場合、EMP テーブルのファイル定義ファイルが emp.jdd というファイル名で生成される。

5.3 環境変数の設定

PostgreSQL インタフェースである libpq ライブラリや 3.5 節で生成した libocdblibrar ライブラリを利用するには環境変数の設定が必要となる。設定対象となる環境変数と、今回設定した値は以下の表の通りである。

PATH	%WORK%¥bin;"C:¥Program Files (x86)¥PostgreSQL¥9.3¥bin";%PATH% (※)環境変数 WORK にはワーキングフォルダを設定している。
COB_PRE_LOAD	libocdblibrary

表 5.3 -1.環境変数の設定内容(1)

次に、COBOL プログラム実行時に参照する環境変数として以下の 4 つがある。

- OC_USERFH：実行時に利用するファイルハンドラ名
- OCDB_DB_NAME：データベース名
- OCDB_DB_USER：データベースユーザ名
- OCDB_DB_PASS：パスワード

最後に COBOL プログラムの連携先をデータベース、ISAM ファイルのどちらかにするかを決定する環境変数を設定する。この環境変数は”テーブル名_DB” という変数名になっており、テーブル単位で設定できる。設定した値が”y”の時はデータベースと連携し、それ以外の値の時は ISAM ファイルと連携する。サンプルプログラムではデータベースの IOTABLE テーブルに接続しているため、今回は環境変数 IOTABLE_DB を設定した。

実際に設定した値は、以下の通り。

OC_USERFH	TESTFH
OCDB_DB_NAME	testdb
OCDB_DB_USER	postgres
OCDB_DB_PASS	postgres
IOTABLE_DB	y

表 5.3-2.環境変数の設定内容(2)

5.4 実行

- INSERTTBL.cbl の実行

cobc コンパイラで生成した INSERTTBL.dll を cobcrun で実行すると、データベース内に IOTABLE テーブルが生成される。この IOTABLE テーブルはファイル定義ファイルを基に生成され、サンプルデータがレコードとして IOTABLE テーブルに登録される。

pgAdmin III で生成された IOTABLE テーブルおよびテーブル内のサンプルデータが登録されていることを確認できる。

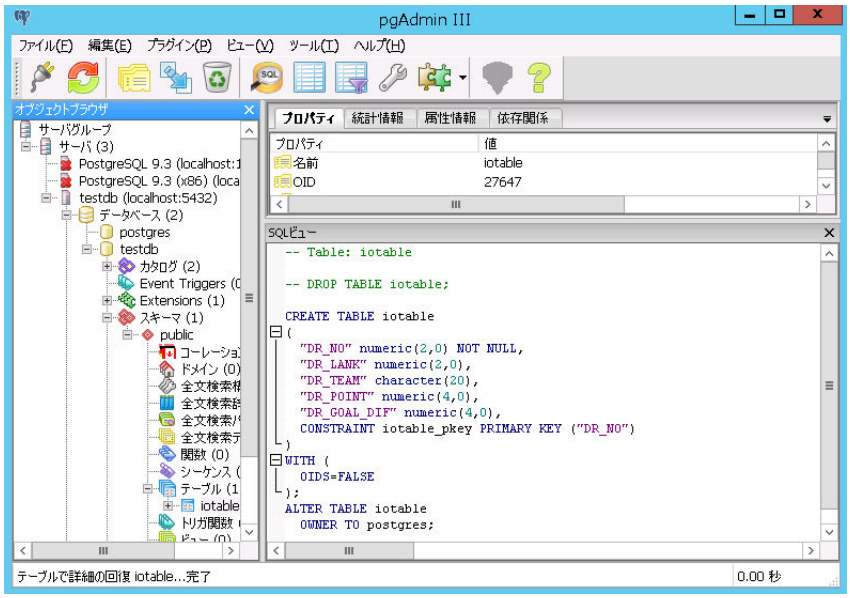
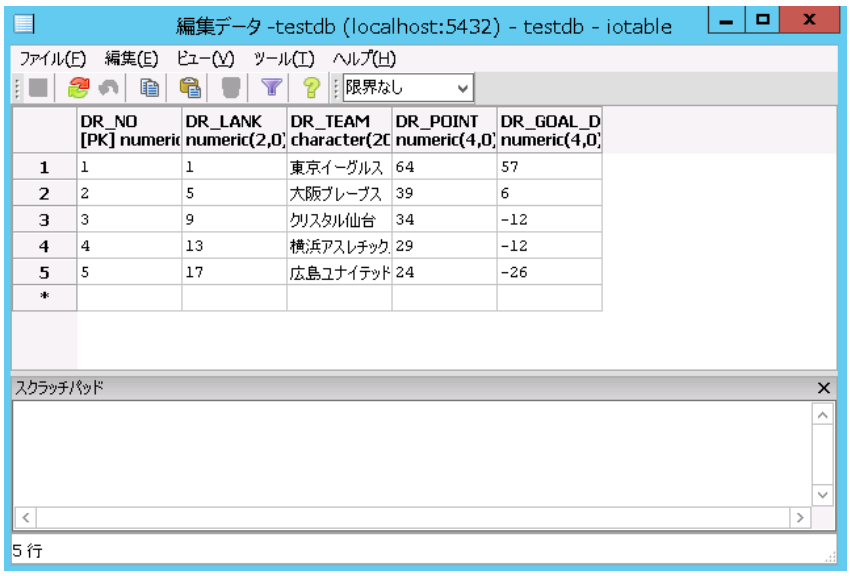


図 5.44.2.2-1. INSERTTBL.cbl 実行後の pgAdmin III 画面



	DR_NO [PK] numeric	DR_LANK numeric(2,0)	DR_TEAM character(20	DR_POINT numeric(4,0)	DR_GOAL_D numeric(4,0)
1	1	1	東京イーグルス	64	57
2	2	5	大阪ブレーブス	39	6
3	3	9	クリスタル仙台	34	-12
4	4	13	横浜アスレチック	29	-12
5	5	17	広島ユナイテッド	24	-26
*					

図 5.44.2.2-2. INSERTTBL.cbl 実行後の IOTABLE のレコード一覧

- FETCHTBL.cbl の実行

cobc コンパイラで生成した FETCHTBL.dll を cobcrun で実行すると、IOTABLE テーブルに登録されているレコードを 1 行ずつフェッチし、ディスプレイ上に表示する。

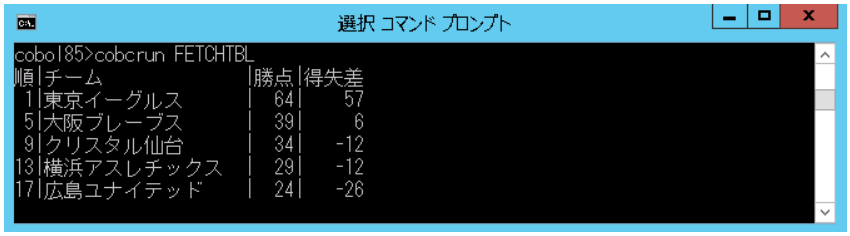


図 5.44.2.2-3. FETCHTBL.cbl 実行結果

5.5 既知の制約事項と問題点

既知の制約事項及び問題点は以下の通り。これらは、今後の機能拡張時の課題とする。

- JDDGenerator
  - COBOL 解析時に COPY 文を参照できない
- Open Cobol DB Library
  - COMP-X など、一部の型に未対応
  - 接続できるデータベースは PostgreSQL のみ
  - 前読み(READ PREVIOUS)に未対応
  - シングル行ロックができない

6 まとめ

opensource COBOL 、データベース連携ツール Open Cobol ESQL、Open Cobol DB Library を利用して、Windows 上で COBOL プログラムとデータベースの連携を確認することができた。これによりオフコン上の COBOL プログラムとデータベースの連携をオープン環境に移行できることが示された。