EDB Postgres Vision Tokyo 2019
リバイバル・ウェビナー

HITACHI
Inspire the Next

日立COBOL + EDB Postgresによる アプリケーション移行検証

2019/11/7

(株)日立製作所　ソフトウェアCoE
OSSソリューションセンタ

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

Contents

1. Oracleからの移行での「壁」

2. 日立COBOL - EDB Postgres評価

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

1. Oracleからの移行での「壁」

- OracleからEDB Postgresへの移行でのノックアウトファクタ
- OracleからEDB Postgresへの移行相談でよくあるシチュエーション
- おさらい：プリプロセッサって何？

© Hitachi, Ltd. 2019. All rights reserved.

1. Oracleからの移行での「壁」

HITACHI
Inspire the Next

■ OracleからEDB Postgresへの移行でのノックアウト・ファクタ

(1)透過暗号化（TDE : Transparent Data Encryption）

金融系など、セキュリティ要件のレベルが高いお客さまには必要な機能ですが、現状、未サポートです。

(2)RACレベルの高可用性、高負荷分散（RAC = Real Application Cluster）

複数インスタンスを並列稼動し、高可用性と高負荷分散を実現する機能。
EDB Postgresでも現状は同等レベルの実現は難しい。
（マルチ・マスターレプリケーションで類似の構成は可能）

(3)COBOL資産

EDB PostgresにはCOBOLのプリプロセッサ(プリコンパイラ)がなく、移行手段がありません！

本日は(3)に関する内容です。

© Hitachi, Ltd. 2019. All rights reserved.

1. Oracleからの移行での「壁」

HITACHI
Inspire the Next

■ OracleからEDB Postgresへの移行相談でよくあるシチュエーション

EDB Postgresの良さをアピールしても、

「COBOL使える？」

この一言で会話が続かなくなるケースがあります。

⬇

なんとかしたい!!

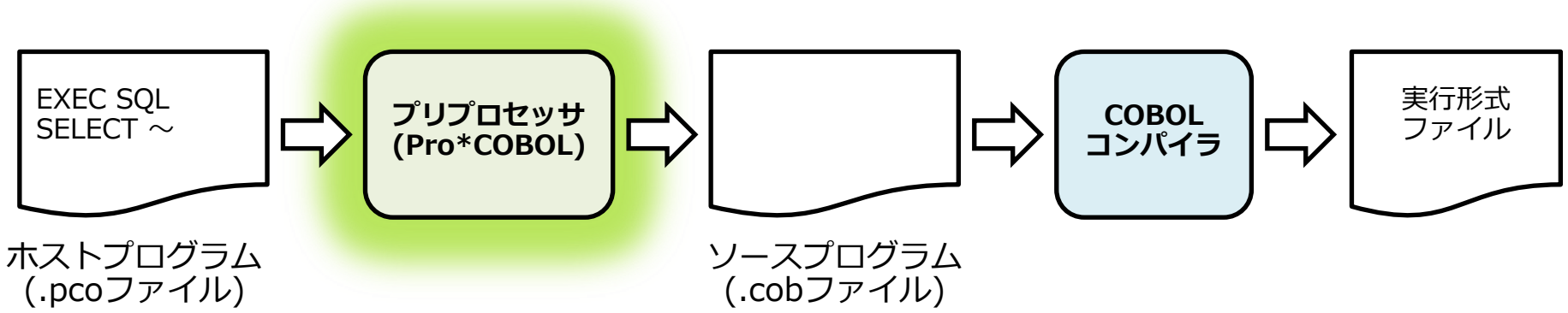
© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

■ おさらい：プロプロセッサって何？

プリプロセッサ(プリコンパイラ)とは、アプリケーションに直接記述されたSQLを解析し、データベースアクセス処理に書き換えるプログラムです。
Oracleでは、Pro*C、Pro*COBOLがこれに該当します。
直接SQLをアプリケーションに記述する形式を「埋め込みSQL」と呼びます。

➢ **Oracle Pro*COBOLでの実行形式ファイル作成までの流れ**



© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

■ おさらい：プロプロセッサって何？

OracleのCOBOL用プリプロセッサPro*COBOLでのプリプロセス前後のソースプログラム例を以下に示します。

プリプロセス前（～.pco）

```
EXEC SQL INSERT INTO TABLE1 VALUES('ABC')
END-EXEC.
```

プリプロセス後（～.cob）

```
        01  SQ0001 GLOBAL.
            02  FILLER PIC X(33) VALUE "insert into TABLE1 values ('ABC')
      -     "".
（省略）
      *     EXEC SQL INSERT INTO TABLE1 VALUES('ABC')
      *     END-EXEC.
            CALL "SQLADR" USING SQ0001 SQL-STMT
            MOVE 1 TO SQL-ITERS
            MOVE 5 TO SQL-OFFSET
            MOVE 0 TO SQL-OCCURS
            CALL "SQLADR" USING
                SQLCUD
                SQL-CUD
            CALL "SQLADR" USING
                SQLCA
                SQL-SQLEST
            MOVE 4352 TO SQL-SQLETY

            CALL "SQLBEX" USING
                SQLCTX
                SQLEXD
                SQLFPN

            IF SQLCODE IN SQLCA IS LESS THAN 0
                THEN PERFORM SQL-ERROR END-IF.
```

SQL文は文字データとなる。

SQLADR関数、SQLBEX関数を呼ぶ処理に変換されています。

弊社HiRDBのCOBOLプリプロセッサも同様に、SQL文を文字データにして実際にSQLを実行する関数を呼ぶよう書き換えます。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

2. 日立COBOL - EDB Postgres連携評価

- ODBCインタフェース機能とは？
- 【評価1】マニュアルベースでの仕様差異調査
- 【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証
- まとめ

© Hitachi, Ltd. 2019. All rights reserved.

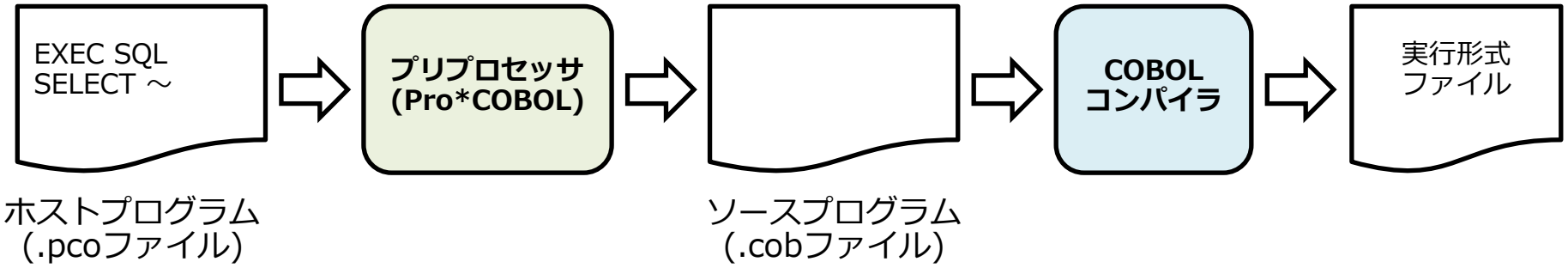
HITACHI
Inspire the Next

■ ODBCインタフェース機能とは？

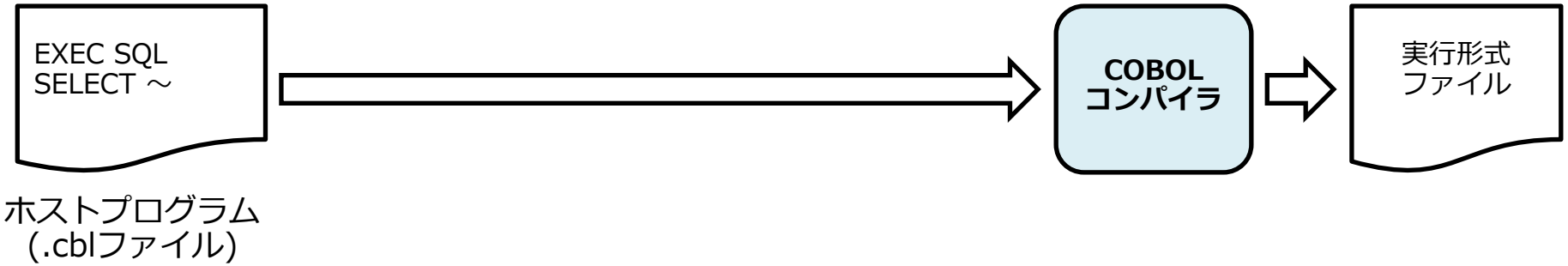
プリプロセッサ（プリコンパイラ）を提供していないデータベース向けに、SQL実行をODBCドライバ経由で実行することでDBアクセスを可能とする**「汎用的なDBアクセス機能」**です。

実行形式ファイル作成までの手順

➢ **Pro*COBOLの場合**



➢ **ODBCインタフェース機能の場合**



© Hitachi, Ltd. 2019. All rights reserved.

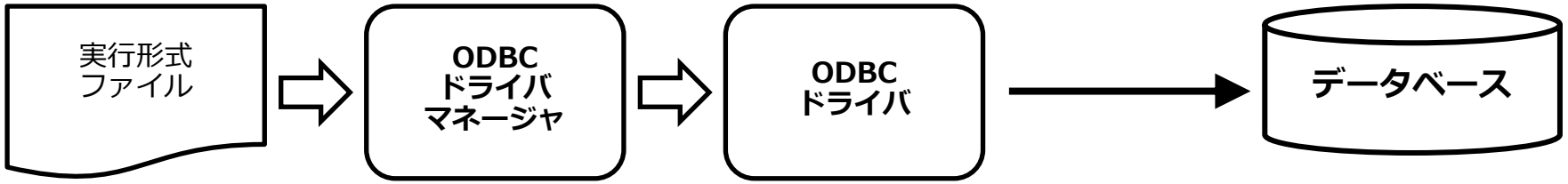
HITACHI
Inspire the Next

■ ODBCインタフェース機能とは？

アプリケーションプログラムからデータベースまでの経路は以下のとおりです。
ODBCドライバをサポートしているDBであれば利用可能です。

➢ Pro*COBOLの場合

➢ ODBCインタフェース機能の場合



凡例 ⇨ 関数呼び出し　→ TCP/IP等の通信

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

はたして、ODBCインタフェース機能でEDB Postgresと連携できるのか？

→ODBCドライバはあるので、前提条件は満たしている。

✓ 「百聞は一見に如かず」ということで、下記評価を実施

【評価1】マニュアルベースでの仕様調査

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価1】マニュアルベースでの仕様調査

マニュアルより、日立COBOLでサポートしている静的SQLは下記。

種類	分類	SQL文	概要
宣言系	埋め込みSQL宣言節	BEGIN DECLARE SECTION	埋め込みSQL開始宣言
		END DECLARE SECTION	埋め込みSQL終了宣言
	埋め込み例外宣言	WHENEVER	埋め込み例外宣言
	カーソル	DECLARE CURSOR	カーソル宣言
制御系	コネクション	CONNECT	コネクションを確立する
		DISCONNECT	コネクションを解除する
		SET CONNECTION	現行コネクションを変更する
	トランザクション	COMMIT	コミットでトランザクションを終了する
		ROLLBACK	ロールバックでトランザクションを終了する

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価1】マニュアルベースでの仕様調査

マニュアルより、日立COBOLでサポートしている静的SQLは下記。（つづき）

種類	分類	SQL文	概要
操作系	静的	SELECT	表の指定された行から値を取り出す
		INSERT	表中に新しい行を作成する
		DELETE	表から行を削除する
		UPDATE	表の行を更新する
	カーソル	OPEN	カーソルを開く
		FETCH	カーソルを表の次の行に位置づけ，その行を取り出す
		CLOSE	カーソルを閉じる
	ストアドプロシジャ	CALL	ストアドプロシージャを呼び出す
	動的	EXECUTE IMMEDIATE	動的SQLを準備し，実行する
		PREPARE	動的SQLを準備する
		DEALLOCATE PREPARE	準備した動的SQLを解放する
		EXECUTE	動的SQLを実行する

静的SQLで実行可能なのは業務使用頻度の多い基本的なSQLのみ。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

✓ 静的SQL･動的SQLについて

弊社HiRDBのマニュアルの説明。

UAPを作成するとき，プログラム中にSQLを記述する方法を**静的SQL**といいます。

これに対し，UAPを作成するとき，プログラム中にSQLを記述しないで，UAP実行時にSQLの文字列を組み立てる方法を**動的SQL**といいます。

※UAP = User Application Program

SQLの内容が実行時の条件に依存しなければ静的SQL、依存するような場合は動的SQLを使用します。

静的SQL	動的SQL

`EXEC SQL INSERT INTO BONUS(ENAME, COMM)`
`  VALUES (:A,:B) END-EXEC.`
	`MOVE "INSERT INTO BONUS(ENAME, COMM) VALUES (:A,:B)"`
	`  TO DYNSTMT-ARR.`
	`MOVE 53 TO DYNSTMT-LEN.`
	`EXEC SQL PREPARE S1 FROM :DYNSTMT END-EXEC.`
	`EXEC SQL EXECUTE S1 USING :ENAME, :COMM END-EXEC.`
	
	この例はSQLを組み立てて無いですが、SQL文を文字として扱うので、プログラム内で任意のSQLを組み立てたり、外部から入力して実行することが可能です。

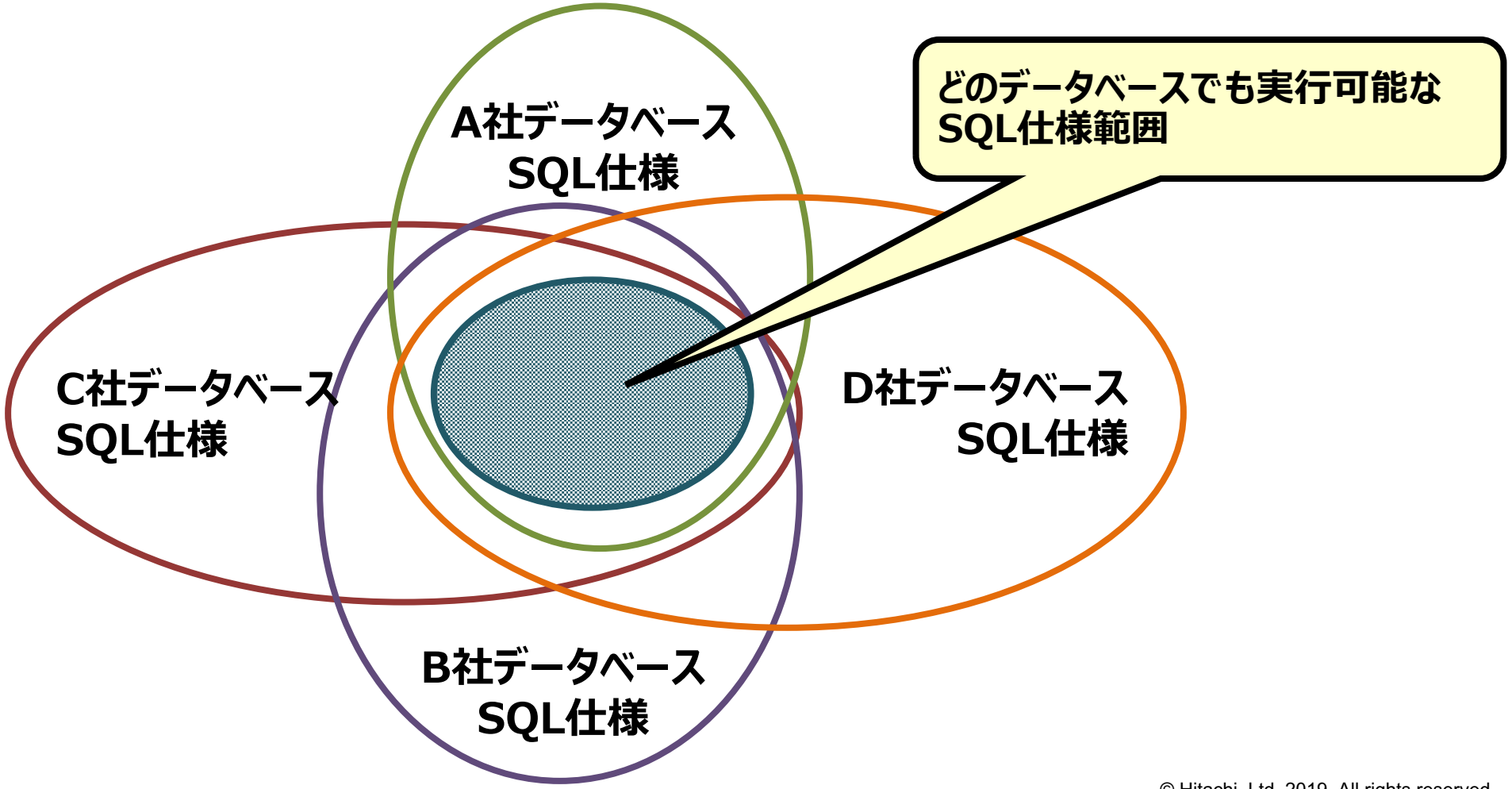
© Hitachi, Ltd. 2019. All rights reserved.

2. 日立COBOL - EDB Postgres連携評価

HITACHI
Inspire the Next

✓ なぜ、静的SQLでは基本的なSQLしか実行できないのか？

静的SQLは構文解析が必要です。しかし、各社データベースのSQLには方言があり、それを意識した構文解析の実装は困難であるため、どのデータベースでも実行できる範囲のSQLのみ実行可能としています。



© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価1】マニュアルベースでの仕様差異調査

静的SQL x Pro*COBOLで実行可能で、静的SQL x 日立COBOLで実行できないSQLを以下に示します。

- 定義系SQL（CREATE TABLE、ALTER TABLE等）
- SQL文宣言（DECLARE STATEMENT）
- 表構造宣言（DECLARE TABLE）
- スレッド機能（ENABLE THREADS）
- 一部の制御系SQL（SET TRANSACTION、SAVE POINT、SET ROLE等）
- 一部の操作系SQL（LOCK TABLE、TRUNCATE TABLE等）
- PL/SQLブロック
- SQL記述子を使用したSQL実行（DESCRIBE）
- LOB操作（LOB OPEN、LOB CLOSE、LOB READ等）

さすがOracle。
機能が豊富です。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価1】マニュアルベースでの仕様差異調査

代表的な構文仕様差異を以下に示します。

種類	Pro*COBOL	日立COBOL
宣言系	WHENEVER { NOT FOUND | NOTFOUND | SQLERROR | SQLWARNING } { CONTINUE | STOP | { GO TO | GOTO } label | STOP | DO PERFORM label | DO CALL "progname" [USING { param } ...] }	WHENEVER { NOT FOUND | SQLERROR | SQLWARNING } { CONTINUE | STOP | { GO TO | GOTO } :label | STOP | PERFORM :label } ※DO PERFORM、DO CALLは記述不可
制御系	COMMIT [WORK] [[COMMENT 'text'] [RELEASE] | FORCE 'text' [, integer]]	COMMIT WORK ※自動切断オプションのRELEASEは指定不可

配列操作ができない等、他に細かいレベルの仕様差異は多々あります。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価1】マニュアルベースでの仕様差異調査

その他、代表的な仕様差異を以下に示します。

分類	Oracle Pro*Cobol	日立COBOL
可変長変数	VARYING	**×：VARYING記述不可**
標識変数	PIC S9(4) COMP	PIC S9(**9**) COMP **桁数の修正が必要です。**
ファイルの引込み	INCLUDE文 〇：宣言文の引込み 〇：SQL文の引込み	COBOL COPY命令 〇：宣言文の引込み **×：SQL文の引込み**
状態変数	SQLSTATE、SQLCODE、SQLCA、ORACA	SQLCODE
FETCH終了時のSQLCODE値	デフォルト1403、プリコンパイラオプションにより100	100
エラーのSQLCODE値	エラーごとの個別の値	-1 **プログラム中での特定のエラーを判定できません。**
警告情報	SQLCAのSQLWARN	+1 **プログラム中での特定の警告を判定できません。**
区切り識別子	〇	**×（識別子を""で囲めません。）**

上記以外にも細かい仕様差異は多々あります。

© Hitachi, Ltd. 2019. All rights reserved.

2. 日立COBOL - EDB Postgres連携評価

HITACHI
Inspire the Next

【評価1】マニュアルベースでの仕様差異調査のまとめ

- 基本的なSQLは静的SQLで問題なく実行できそう
- 静的SQLで実行できないSQLは、動的SQLで実行できそう
- 可変長変数、SQLCODEの値など、仕様差異には致命的なものはなく、対処は可能

↓

でも、実際はどうか???

Pro*COBOLアプリケーションを移行し、
実際に直面しそうな移行課題を検証。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

Pro*COBOLの機能を広く確認する目的で、Pro*COBOLのサンプルアプリケーションを日立COBOLに移行する検証を実施。

本日は一例として単純な問合せを実行するPro*COBOLのサンプルプログラム、sample1.pcoの移行例を説明します。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

sample1.pco（単純な問合せ）の移行（1/5）

修正前

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. QUERY.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.

           EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01  USERNAME          PIC X(10) VARYING.
       01  PASSWD            PIC X(10) VARYING.

       01  EMP-REC-VARS.
           05  EMP-NAME      PIC X(10) VARYING.


           05  EMP-NUMBER    PIC S9(4) COMP VALUE ZERO.
           05  SALARY        PIC S9(5)V99 COMP-3 VALUE ZERO.

           05  COMMISSION    PIC S9(5)V99 COMP-3 VALUE ZERO.

           05  COMM-IND      PIC S9(4) COMP VALUE ZERO.
           EXEC SQL END DECLARE SECTION END-EXEC.

           EXEC SQL INCLUDE SQLCA END-EXEC.
```

修正後

```
ID DIVISION.
       PROGRAM-ID. QUERY.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.

           EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01  USERNAME          PIC X(10).
       01  PASSWD            PIC X(10).
       01  ODBC-DSN          PIC X(10).
       01  EMP-REC-VARS.
           0   EMP-NAME.
               EMP-NAME-LEN PIC S9(9) COMP.
               NAME-ARR PIC X(10).
```

- 「VARYING」はPro*COBOL用のキーワードでPro*COBOLが集団項目に変換。
  日立COBOLではVARYINGを記述できないため、固定長に変更。
- CONNECTで指定するユーザ名及びパスワードの変数宣言に加えてODBCのデータソース名の変数宣言を追加。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

sample1.pco（単純な問合せ）の移行（1/5）

修正前

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. QUERY.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.

           EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01  USERNAME          PIC X(10) VARYING.
       01  PASSWD            PIC X(10) VARYING.

       01  EMP-REC-VARS.
           05  EMP-NAME      PIC X(10) VARYING.


           05  EMP-NUMBER    PIC S9(4) COMP VALUE ZERO.
           05  SALARY        PIC S9(5)V99 COMP-3 VALUE ZERO.

           05  COMMISSION    PIC S9(5)V99 COMP-3 VALUE ZERO.

           05  COMM-IND      PIC S9(4) COMP VALUE ZERO.
           EXEC SQL END DECLARE SECTION END-EXEC.

           EXEC SQL INCLUDE SQLCA END-EXEC.
```

修正後

```
ID DIVISION.
```

日立COBOLではVARYING使用不可。
こちらの例では集団項目での宣言に変更。
このとき、基本項目の変数名はPro*COBOLによる展開形に合わせる。
（XXX-LEN、XXX-ARR）

```
       01  PA                 PIC X(10).
       01  ODE                PIC X(10).
       01  EMP REC-VARS.
           05  EMP-NAME.
               10 EMP-NAME-LEN PIC S9(9) COMP.
               10 EMP-NAME-ARR PIC X(10).
           05  EMP-NUMBER     PIC S9(4) COMP VALUE ZERO.
           05  SALARY         PIC S9(5)V99
                 DISPLAY SIGN LEADING SEPARATE VALUE ZERO.
           05  COMMISSION     PIC S9(5)V99
                 DISPLAY SIGN LEADING SEPARATE VALUE ZERO.
           05  COMM-IND       PIC S9(9) COMP VALUE ZERO.
           EXEC SQL END DECLARE SECTION END-EXEC.
```

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

✓ Pro*COBOLでの可変長データ型を示す「VARYING」について

VARYINGを付与した変数は、Pro*COBOLにより集団項目に変換されます。

部位	プリプロセス前（～.pco）	プリプロセス後（～.cob）

宣言	`01 USERNAME PIC X(10) VARYING.`
	`01 PASSWD PIC X(10) VARYING.`
		`*01 USERNAME PIC X(10) VARYING.`
		`01 USERNAME.`
		`02 USERNAME-LEN PIC S9(4) COMP-5.`
		`02 USERNAME-ARR PIC X(10).`
		`*01 PASSWD PIC X(10) VARYING.`
		`01 PASSWD.`
		`02 PASSWD-LEN PIC S9(4) COMP-5.`
		`02 PASSWD-ARR PIC X(10).`

使用箇所
集団項目として使用
	`LOGON.`
	`MOVE "user1" TO USERNAME-ARR.`
	`MOVE 5 TO USERNAME-LEN.`
	`MOVE "user1" TO PASSWD-ARR.`
	`MOVE 5 TO PASSWD-LEN.`
		`LOGON.`
		`MOVE "user1" TO USERNAME-ARR.`
		`MOVE 5 TO USERNAME-LEN.`
		`MOVE "user1" TO PASSWD-ARR.`
		`MOVE 5 TO PASSWD-LEN.`

同じ

変数の使用箇所では、Pro*COBOLが変換後の集団項目として使用する必要があります。
したがって、VARYINGを付与した変数は、Pro*COBOLの変換後と同じ名称の集団項目にすることで、使用箇所の修正が不要となります。
元の変数名の末尾に-LEN、-ARRを付与すれば良いです。
（加えて、XXX-LENの桁数を4→9に変更要。）

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

sample1.pco（単純な問合せ）の移行（1/5）

修正前

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. QUERY.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.

           EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01  USERNAME          PIC X(10) VARYING.
       01  PASSWD            PIC X(10) VARYING.

       01  EMP-REC-VARS.
           05  EMP-NAME      PIC X(10) VARYING.


           05  EMP-NUMBER    PIC S9(4) COMP VALUE ZERO.
           05  SALARY        PIC S9(5)V99 COMP-3 VALUE ZERO.

           05  COMMISSION    PIC S9(5)V99 COMP-3 VALUE ZERO.

           05  COMM-IND      PIC S9(4) COMP VALUE ZERO.
           EXEC SQL END DECLARE SECTION END-EXEC.

           EXEC SQL INCLUDE SQLCA END-EXEC.
```

修正後

```
ID DIVISION.
       PROGRAM-ID. QUERY.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.

           EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01  USERNAME          PIC X(10).
       01  PASSWD            PIC X(10).
```

ODBCインタフェース機能では、COMP-3は使用できないため、DISPLAY SIGN LEADING SEPARATEに修正。

```
           10 [illegible]ME-ARR PIC X(10).
           05  EMP-NUMBER    PIC S9(4) COMP VALUE ZERO.
           05  SALARY        PIC S9(5)V99
               DISPLAY SIGN LEADING SEPARATE VALUE ZERO.
           05  COMMISSION    PIC S9(5)V99
               DISPLAY SIGN LEADING SEPARATE VALUE ZERO.
           05  COMM-IND      PIC S9(9) COMP VALUE ZERO.
           EXEC SQL END DECLARE SECTION END-EXEC.

           EXEC SQL INCLUDE SQLCA END-EXEC.
```

標識変数の桁数を修正

SQLCAは使えないので削除

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

sample1.pco（単純な問合せ）の移行（2/5）

修正前

```
01  DISPLAY-VARIABLES.
    05  D-EMP-NAME     PIC X(10).
    05  D-SALARY       PIC Z(4)9.99.
    05  D-COMMISSION   PIC Z(4)9.99.
    05  D-EMP-NUMBER   PIC 9(4).

01 D-TOTAL-QUERIED   PIC 9(4) VALUE ZERO.

PROCEDURE DIVISION.
BEGIN-PGM.
    EXEC SQL WHENEVER SQLERROR
       DO PERFORM SQL-ERROR END-EXEC.

    PERFORM LOGON.
QUERY-LOOP.
    DISPLAY " ".
    DISPLAY "ENTER EMP NUMBER (0 TO QUIT): "
        WITH NO ADVANCING.

    ACCEPT D-EMP-NUMBER.
    MOVE D-EMP-NUMBER TO EMP-NUMBER.
    IF (EMP-NUMBER = 0)
        PERFORM SIGN-OFF.
    MOVE SPACES TO EMP-NAME-ARR.
    EXEC SQL WHENEVER NOT FOUND GOTO NO-EMP END-EXEC.
```

修正後

```
01  DISPLAY-VARIABLES.
    05  D-EMP-NAME     PIC X(10).
    05  D-SALARY       PIC Z(4)9.99.
    05  D-COMMISSION   PIC Z(4)9.99.
    05  D-EMP-NUMBER   PIC 9(4).

01 D-TOTAL-QUERIED   PIC 9(4) VALUE ZERO.

PROCEDURE DIVISION.
BEGIN-PGM.
    EXEC SQL WHENEVER SQLERROR
       PERFORM :SQL-ERROR END-EXEC.

    PERFORM LOGON.
QUERY-LOOP.
    DISPLAY " ".
    DISPLAY "ENTER EMP NUMBER (0 TO QUIT): "
        WITH NO ADVANCING.

    ACCEPT D-EMP-NUMBER.
    MOVE D-EMP-NUMBER TO EMP-NUMBER.
    IF (EMP-NUMBER = 0)
        PERFORM SIGN-OFF.
    MOVE SPACES TO EMP-NAME-ARR.
    EXEC SQL WHENEVER NOT FOUND GOTO :NO-EMP END-EXEC.
```

DOを削除し，ホスト識別子の前にコロン追加

ホスト識別子の前にコロン追加

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

sample1.pco（単純な問合せ）の移行（3/5）

修正前

```
        EXEC SQL SELECT ENAME, SAL, NVL(COMM, 0)
                INTO :EMP-NAME, :SALARY, :COMMISSION:COMM-IND
                FROM EMP
                WHERE EMPNO = :EMP-NUMBER
            END-EXEC.
            PERFORM DISPLAY-INFO.
            ADD 1 TO D-TOTAL-QUERIED.
            GO TO QUERY-LOOP.

        NO-EMP.
            DISPLAY "NOT A VALID EMPLOYEE NUMBER - TRY
AGAIN.".
            GO TO QUERY-LOOP.

        LOGON.
            MOVE "user1" TO USERNAME-ARR.
            MOVE 5 TO USERNAME-LEN.
            MOVE "user1" TO PASSWD-ARR.
            MOVE 5 TO PASSWD-LEN.
            EXEC SQL
                CONNECT :USERNAME IDENTIFIED BY :PASSWD

            END-EXEC.
            DISPLAY " ".
            DISPLAY "CONNECTED TO ORACLE AS USER: ",
USERNAME-ARR.
```

修正後

```
        EXEC SQL SELECT ENAME, SAL, COMM
                INTO :EMP-NAME, :SALARY, :COMMISSION:COMM-IND
                FROM EMP
                WHERE EMPNO = :EMP-NUMBER
            END-EXEC.
            PERFORM DISPLAY-INFO.
            ADD 1 TO D-TOTAL-QUERIED.
            GO TO QUERY-LOOP.
                                                        ".

            END-EXEC.
            DISPLAY " ".
            DISPLAY "CONNECTED TO EDB(ODBC) AS USER: ",
USERNAME.
```

NVL関数でコンパイルエラー発生!!
日立COBOLでのSQL解析は基本的なSQLのみ受理可能です。
このサンプルではNULL値判定にNVL関数を使っているので標識変数での判定に変更。

※NVL関数とは？
第1引数がNULL値の場合、第2引数を返却

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

sample1.pco（単純な問合せ）の移行（3/5）

修正前

```
EXEC SQL SELECT ENAME, SAL, NVL(COMM, 0)
        INTO :EMP-NAME, :SALARY, :COMMISSION:COMM-IND
        FROM EMP
        WHERE EMPNO = :EMP-NUMBER
    END-EXEC.
    PERFORM DISPLAY-INFO.
    ADD 1 TO D-TOTAL-QUERIED.
    GO TO QUERY-LOOP.

NO-EMP.
    DISPLAY
    "NOT A VALID EMPLOYEE NUMBER - TRY AGAIN.".
    GO TO QUERY-LOOP.

LOGON.
    MOVE "user1" TO USERNAME-ARR.
    MOVE 5 TO USERNAME-LEN.
    MOVE "xxxxx" TO PASSWD-ARR.
    MOVE 5 TO PASSWD-LEN.
    EXEC SQL
        CONNECT :USERNAME IDENTIFIED BY :PASSWD

    END-EXEC.
    DISPLAY " ".
    DISPLAY "CONNECTED TO ORACLE AS USER: ",
        USERNAME-ARR.
```

修正後

```
EXEC SQL SELECT ENAME, SAL, COMM
        INTO :EMP-NAME, :SALARY, :COMMISSION:COMM-IND
        FROM EMP
        WHERE EMPNO = :EMP-NUMBER
    END-EXEC.
    PERFORM DISPLAY-INFO.

LOGON.
    MOVE "user1" TO USERNAME.
    MOVE "xxxxx" TO PASSWD.
    MOVE "edb1" TO ODBC-DSN.

    EXEC SQL
        CONNECT :USERNAME IDENTIFIED BY :PASSWD
            USING :ODBC-DSN
    END-EXEC.
    DISPLAY " ".
    DISPLAY "CONNECTED TO EDB(ODBC) AS USER: ",
        USERNAME.
```

- CONNECTで指定するユーザ名及びパスワードのホスト変数を固定長変数に修正したので、値を設定する変数名を修正。
- 固定長変数なので長さの設定が不要になり削除。
- ODBCのデータソース名の設定処理を追加。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

sample1.pco（単純な問合せ）の移行（4/5）

修正前

```
DISPLAY-INFO.
    DISPLAY " ".
    DISPLAY "EMPLOYEE    SALARY    COMMISSION".
    DISPLAY "--------    ------    ----------".
    MOVE EMP-NAME-ARR TO D-EMP-NAME.
    MOVE SALARY TO D-SALARY.
    IF COMM-IND = -1
        DISPLAY D-EMP-NAME, D-SALARY,
                "          NULL"
    ELSE
        MOVE COMMISSION TO D-COMMISSION
        DISPLAY D-EMP-NAME, D-SALARY, "      ",
                D-COMMISSION
    END-IF.

SIGN-OFF.
    DISPLAY " ".
    DISPLAY "TOTAL NUMBER QUERIED WAS ",
        D-TOTAL-QUERIED, ".".
    DISPLAY " ".
    DISPLAY "HAVE A GOOD DAY.".
    DISPLAY " ".
    EXEC SQL COMMIT WORK RELEASE END-EXEC.

    STOP RUN.
```

修正後

```
DISPLAY-INFO.
    DISPLAY " ".
    DISPLAY "EMPLOYEE    SALARY    COMMISSION".
    DISPLAY "--------    ------    ----------".
    MOVE EMP-NAME-ARR TO D-EMP-NAME.
    MOVE SALARY TO D-SALARY.
    IF COMM-IND = -1
        DISPLAY D-EMP-NAME, D-SALARY,
        "          NULL"
    ELSE
        MOVE COMMISSION TO D-COMMISSION
        DISPLAY D-EMP-NAME, D-SALARY, "      ",
        D-COMMISSION
    END-IF.

SIGN-OFF.
    DISPLAY
    DISPLAY
        D-T
    DISPLAY
    DISPLAY "HAVE  OOD DAY.".
    DISPLAY " ".
    EXEC SQL COMMIT WORK END-EXEC.
    EXEC SQL DISCONNECT END-EXEC.
    STOP RUN.
```

COMMIT RELASEは未サポートであるため、COMMITとDISCONNECTに修正。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

sample1.pco（単純な問合せ）の移行（5/5）

修正前

```
SQL-ERROR.
    EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.
    DISPLAY " ".
    DISPLAY "ORACLE ERROR DETECTED:".
    DISPLAY " ".
    DISPLAY SQLERRMC.
    EXEC SQL ROLLBACK WORK RELEASE END-EXEC.

    STOP RUN.
```

修正後

```
SQL-ERROR.
    EXEC S                                  C.
    DISPLA
    DISPLAY "  ED                ERROR DETECTED:".
    DISPLAY "  ".
    DISPLAY SQLERRMC.
    EXEC SQL ROLLBACK WORK END-EXEC.
    EXEC SQL DISCONNECT END-EXEC.
    STOP RUN.
```

エラー時は日立COBOLがメッセージを出力するのでSQLERRMCの出力を削除。

ROLLBACK RELEASEは未サポートであるため、ROLLBACKとDISCONNECTに修正。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

✓ NVL関数等、COBOLがエラーとするキーワードの対応について

先ほどの修正例では、NVL関数なしで回避できましたが、ほとんどの場合は関数をそのまま使う必要があると考えます。
日立COBOLのODBCインタフェース機能では、関数はMAX、AVGなど基本的な集合関数のみ記述可能であり、他の関数はエラーになるケースが多いです。
その場合は、動的SQLとして実行し、日立COBOLによる構文解析を回避します。

修正前	修正後
EXEC SQL SELECT ENAME, SAL, NVL(COMM, 0) INTO :EMP-NAME, :SALARY, :COMMISSION :COMM-IND FROM EMP WHERE EMPNO = :EMP-NUMBER END-EXEC.	EXEC SQL PREPARE STMT1 FROM 'SELECT ENAME, SAL, NVL(COMM, 0) FROM EMP '- 'WHERE EMPNO = ?' END-EXEC. EXEC SQL EXECUTE STMT1 USING :EMP-NUMBER INTO :EMP-NAME, :SALARY, :COMMISSION :COMM-IND END-EXEC.

?パラメタに与える値

検索結果を受け取る変数

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

■ その他代表的な修正例

✓ 日立COBOLの静的SQLで実行できないSQL文は動的SQLで実行する

修正前	修正後
EXEC SQL CREATE TABLE OUTLNDEMO_TAB AS SELECT EMPNO, ENAME, SAL FROM EMP WHERE DEPTNO = 10 END-EXEC.	EXEC SQL EXECUTE IMMEDIATE 'CREATE TABLE OUTLNDEMO_TAB AS '- 'SELECT EMPNO, ENAME, SAL FROM EMP ’- 'WHERE DEPTNO = 10' END-EXEC.
EXEC SQL DROP TABLE OUTLNDEMO_TAB END-EXEC.	EXEC SQL EXECUTE IMMEDIATE 'DROP TABLE OUTLNDEMO_TAB' END-EXEC.

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証

【調査中】CALL文

CALL文実行時、COBOLでのエラー、ODBCドライバ内で異常終了などが発生。
現在、調査中ですが、SELECTへの書換えで回避は可能です。

修正前	修正後

EXEC SQL EXECUTE
BEGIN
FUNC1(:PARAM1, :PARAM2, :PARAM3);
END;
END-EXEC.
	EXEC SQL PREPARE STMT1 FROM
	'SELECT * FROM FUNC1(?,?,?)'
	END-EXEC.
	EXEC SQL EXECUTE STMT1
	USING :PARAM1, :PARAM2, :PARAM3
	END-EXEC.

【評価2】Pro*COBOLのサンプルアプリケーションによる実機検証まとめ

評価したサンプルプログラム14個は、すべて動作するよう書換えは可能。
（CALL文は上記のとおり、SELECTへの書換えで対応）

© Hitachi, Ltd. 2019. All rights reserved.

2. 日立COBOL - EDB Postgres連携評価

HITACHI
Inspire the Next

【まとめ】ドキュメント、および実機評価の結果をまとめます。

- ✓ 基本的なSQLは静的SQLで実行可能
- ✓ 日立COBOLで未サポートの機能を使用したSQLは、動的SQLにすることで実行可能
- ✓ その他仕様差異では致命的なものは無い。
- ✓ CALL文は継続調査中ですが、回避策はある。

↓

静的SQLで実行できる範囲が少なく、ほぼ動的SQLへの書換えが必要。
ただし、修正内容はパターン化でき、他言語への書換えよりはコストが低減できそう。

↓

COBOLアプリケーションも移行対象として十分検討の余地があると考えます！

© Hitachi, Ltd. 2019. All rights reserved.

他社所有商標に関する表示

HITACHI
Inspire the Next

- PostgreSQLは、PostgreSQL Community Association of Canadaのカナダにおける登録商標およびその他の国における商標です。
- EnterpriseDBは、EnterpriseDB Corporationの登録商標です。EDBおよびEDB Postgresは、EnterpriseDB Corporationの商標です。
- Oracleは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。
- その他記載の会社名、製品名などは、それぞれの会社の商標もしくは登録商標です。

© Hitachi, Ltd. 2019. All rights reserved.

HITACHI
Inspire the Next

END

日立COBOL + EDB Postgresによる アプリケーション移行検証

2019/11/7
(株)日立製作所　ソフトウェアCoE
OSSソリューションセンタ

© Hitachi, Ltd. 2019. All rights reserved.