



EDB Postgres™ Advanced Server OCL Connector Guide

リリース **12.1.2.1**

EnterpriseDB

2019 年 12 月 15 日

目次

第 1 章	新機能	3
第 2 章	サポートされているプラットフォーム	5
第 3 章	Open Client Library	7
3.1	OCL コネクタのインストールと設定	8
3.1.1	RPM パッケージを使用したコネクタのインストール	8
3.1.2	SLES 12 ホストへのコネクタのインストール	10
3.1.3	Debian または Ubuntu ホストへの DEB パッケージのインストール	12
3.1.4	グラフィカルインストーラを使用してコネクタをインストールする	13
3.2	接続文字列の作成	15
3.3	プログラムのコンパイルとリンク	19
3.4	Ref Cursor Support	20
3.5	OCL 関数リファレンス	23
3.5.1	関数の接続、承認、初期化	23
3.5.2	ハンドルおよび記述子関数	25
3.5.3	関数のバインド、定義、および説明	29
3.5.4	ステートメント関数	29
3.5.5	トランザクション関数	29
3.5.6	XA 関数	30
3.5.7	日付および日時関数	31
3.5.8	インターバル関数	32
3.5.9	数字関数	33
3.5.10	文字列関数	35
3.5.11	カートリッジサービスとファイル I/O インターフェイス関数	35
3.5.12	LOB 関数	35
3.5.13	その他の機能	36
3.5.14	サポートされているデータ型	37
3.6	OCL エラーコード-リファレンス	38
3.7	マルチスレッドのサポート	40
第 4 章	OCL トレースの生成	41

第 5 章	SSL の使用	43
第 6 章	結論	45
索引		47

OCL コネクタは、Oracle Call Interface に似た API を提供します。Oracle Call Interface を使用するように作成されたアプリケーションは、Enterprise Server の OCL コネクタを使用して Advanced Server データベースサーバーとやり取りするために再コンパイルできます。

このガイドでは、以下に関するインストールおよび使用手順を説明します。

- コネクタのインストール方法。
- Oracle スタイルの接続文字列の形成方法。
- プログラムをコンパイルおよびリンクする方法。

このガイドには、Advanced Server でサポートされている機能のリファレンスセクションも含まれています。

注釈: EnterpriseDB は、Oracle Real Application Clusters (RAC) および Oracle Exadata での Open Client Library の使用をサポートしていません。前述の Oracle 製品は、この EnterpriseDB 製品では評価も認定もされていません。

第 1 章

新機能

Advanced Server OCL Connector 12.1.2.1 を作成するために、次の機能が追加されました。

- EDB OCL Connector はマルチスレッドのサポートを提供するようになりました。
- EDB OCL コネクタが EDB Postgres Advanced Server 12 をサポートするようになりました。
- EDB OCL コネクタは、Windows Server 2019 プラットフォームでもサポートされるようになりました。

第 2 章

サポートされているプラットフォーム

Advanced Server OCL Connector は、Advanced Server バージョン 9.4 以降で認定されています。OCL コネクタネイティブパッケージは、次の 64 ビットプラットフォームでサポートされています。

- Red Hat Enterprise Linux (x86_64) 6.x および 7.x
- CentOS (x86_64) 6.x および 7.x
- OEL Linux 6.x および 7.x
- RHEL または CentOS 7.x を実行している PPC-LE 8
- SLES 12.x
- Debian 9.x
- Ubuntu 18.04

OCL コネクタのグラフィカルインストーラーは、次の 64 ビット Windows プラットフォームでサポートされています。

- Windows Server 2019
- Windows Server 2016
- Windows Server 2012 R2
- Windows 10
- Windows 8
- Windows 7

OCL コネクタのグラフィカルインストーラーは、次の 32 ビット Windows プラットフォームでサポートされています。

- Windows 10

- Windows 8
- Windows 7

第 3 章

Open Client Library

Open Client Library は、Oracle Call Interface とのアプリケーションの相互運用性を提供します。以前はロックされていたアプリケーションは、アプリケーションコードを最小限またはまったく変更することなく、EDB Postgres Advanced Server または Oracle データベースで動作できるようになりました。

次の図は、Open Client Library と Oracle Call Interface アプリケーションスタックを比較しています。

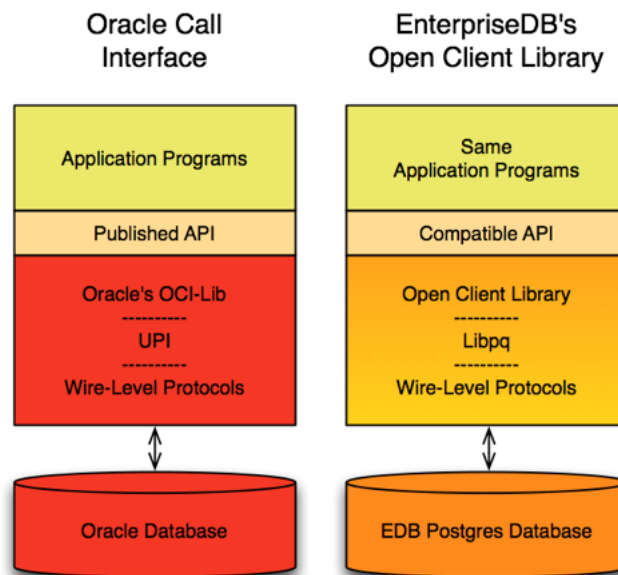


図 1 *Comparison with Oracle Call Interface*

Open Client Library の EnterpriseDB 実装は C で記述されています。

3.1 OCL コネクタのインストールと設定

RPM パッケージ、ネイティブパッケージ、またはグラフィカルインストーラーを使用して、OCL コネクタをインストールまたは更新できます。

3.1.1 RPM パッケージを使用したコネクタのインストール

OCL Connector をインストールする前に、以下を行う必要があります。

epel-release パッケージをインストールします:

```
yum -y install https://dl.fedoraproject.org/pub/epel/
epel-release-latest-7.noarch.rpm
```

注釈: CentOS-Base.repo ファイル (/etc/yum.repos.d にある) で [extras] リポジトリ定義を有効にする必要があるかもしれません。

EnterpriseDB リポジトリへのアクセスを許可する資格情報も必要です。資格情報のリクエストについては、次をご覧ください。

<https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf>

リポジトリの資格情報を受け取った後、次のことができます。

1. リポジトリ構成ファイルを作成します。
2. ユーザー名とパスワードを指定して、ファイルを変更します。
3. edb-oci をインストールします。

リポジトリ設定ファイルの作成

リポジトリ構成ファイルを作成するには、スーパーユーザー特権を引き受けて、次のコマンドを呼び出します。

```
yum -y install https://yum.enterprisedb.com/edb-repo-rpms/
edb-repo-latest.noarch.rpm
```

リポジトリ設定ファイルの名前は edb.repo です。ファイルは /etc/yum.repos.d にあります。

ファイルの変更、ユーザー名とパスワードの入力

edb.repo ファイルを作成した後、エディターの選択を使用して、有効なパラメーターの値が 1 であることを確認し、edb.repo の username および password プレースホルダーを置き換えます 4 '' 登録済みの EnterpriseDB ユーザーの名前とパスワードを含む仕様。

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-
↪$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

OCL コネクタのインストール

設定ファイルへの変更を保存した後、`yum install` コマンドを使用して OCL Connector をインストールできます。たとえば、次のコマンドは OCL Connector をインストールします。

```
yum install edb-oci

yum install edb-oci-devel
```

システムで認識されないソースによって署名された RPM パッケージをインストールすると、`yum` はローカルサーバーにキーをインポートする許可を求める場合があります。プロンプトが表示され、パッケージが信頼できるソースからのものであることに満足したら、`y` を入力し、Return を押して続行します。

インストール中に、`yum` は解決できない依存関係に遭遇する場合があります。存在する場合、手動で解決する必要がある依存関係のリストが提供されます。

RPM インストールの更新

既存の OCL Connector RPM インストールがある場合、`yum` を使用してリポジトリ設定ファイルをアップグレードし、より新しい製品バージョンに更新できます。edb.repo ファイルを更新するには、スーパーユーザー権限を引き受けて次のように入力します：

```
yum upgrade edb-repo
```

`yum` は edb.repo ファイルを更新して、現在の EDB リポジトリへのアクセスを有効にし、edb.repo ファイルで指定された資格情報で接続するように設定します。次に、`yum` を使用して、インストールされているパッケージをアップグレードできます。

```
yum install edb-oci

yum install edb-oci-devel
```

3.1.2 SLES 12 ホストへのコネクタのインストール

zypper パッケージマネージャーを使用して、SLES 12 ホストにコネクタをインストールできます。zypper は、パッケージをインストールするときにパッケージの依存関係を満たそうとしますが、EnterpriseDB でホストされていない特定のリポジトリへのアクセスが必要です。コネクタをインストールする前に、次のコマンドを使用して EnterpriseDB リポジトリ構成ファイルを SLES ホストに追加します。

```
zypper addrepo https://zypp.enterprisedb.com/suse/epas12-sles.repo
zypper addrepo https://zypp.enterprisedb.com/suse/epas-sles-tools.repo
zypper addrepo
https://zypp.enterprisedb.com/suse/epas-sles-dependencies.repo
```

各コマンドは、/etc/zypp/repos.d ディレクトリにリポジトリ設定ファイルを作成します。ファイルの名前は次のとおりです。

- edbas12suse.repo
- edbasdependencies.repo
- edbastools.repo

リポジトリ設定ファイルを作成したら、zypper refresh コマンドを使用して SLES ホストのメタデータを更新し、EnterpriseDB リポジトリを含めます。

```
/etc/zypp/repos.d # zypper refresh
Repository 'SLES12-12-0' is up to date.
Repository 'SLES12-Pool' is up to date.
Repository 'SLES12-Updates' is up to date.
Retrieving repository 'EDB Postgres Advanced Server 12 12 - x86_64'
metadata -----[\]

Authentication required for
'https://zypp.enterprisedb.com/12/suse/suse-12-x86_64'
User Name:
Password:

Retrieving repository 'EDB Postgres Advanced Server 12 12 - x86_64'
metadata.....[done]
Building repository 'EDB Postgres Advanced Server 12 12 - x86_64'
cache.....[done]
All repositories have been refreshed.
...
```

User Name および Password の入力を求められたら、EnterpriseDB リポジトリの接続認証情報を入力します。資格情報が必要な場合は、**“EnterpriseDB <<https://www.enterprisedb.com/repository-access-request/>>”**。

EDB Postgres Advanced Server またはサポートコンポーネントをインストールする前に、SUSEConnect と SUSE

Package Hub 拡張機能を SLES ホストに追加し、ホストを SUSE に登録して、SUSE リポジトリへのアクセスを許可する必要があります。次のコマンドを使用します。

```
zypper install SUSEConnect
SUSEConnect -p PackageHub/12/x86_64
SUSEConnect -p sle-sdk/12/x86_64
```

SUSE ホストの登録の詳細については、“SUSE Web サイト<<https://www.suse.com/support/kb/doc/?id=7016626>>’_。

次に、zypper ユーティリティを使用してコネクタをインストールできます。

```
zypper install edb-oci
zypper install edb-oci-devel
```

3.1.3 Debian または Ubuntu ホストへの DEB パッケージのインストール

Debian または Ubuntu ホストに DEB パッケージをインストールするには、EnterpriseDB リポジトリへのアクセスを許可する資格情報が必要です。リポジトリの資格情報を要求するには、**“EDB Web サイトにアクセスしてください”**<<https://www.enterprisedb.com/repository-access-request>>‘_。

次の手順では、EnterpriseDB apt リポジトリを使用して DEB パッケージをインストールする手順を説明します。コマンドを使用する場合、username と password を EnterpriseDB が提供する資格情報に置き換えます。

1. スーパーユーザー特権を想定します。

```
sudo su -
```

2. EnterpriseDB リポジトリを構成します。

```
sh -c 'echo "deb
https://username:password@apt.enterprisedb.com/${lsb_release
-cs)-edb/ ${lsb_release -cs) main" >
/etc/apt/sources.list.d/edb-${lsb_release -cs).list'
```

3. 安全な APT リポジトリのサポートをシステムに追加します。

```
apt-get install apt-transport-https
```

4. EBD 署名キーを追加します。

```
wget -q -O - https://username:password
@apt.enterprisedb.com/edb-deb.gpg.key | apt-key add -
```

5. リポジトリのメタデータを更新します。

```
apt-get update
```

6. DEB パッケージをインストールします。

```
apt-get install edb-oci
```

```
apt-get install edb-oci-dev
```

3.1.4 グラフィカルインストーラを使用してコネクタをインストールする

EnterpriseDB コネクタインストールウィザードを使用して、システムに OCL コネクタを追加できます。ウィザードは “EnterpriseDB Advanced Downloads” ページから入手できます(<<https://www.enterprisedb.com/advanced-downloads>>)。

このセクションでは、インストールウィザードを使用して Windows システムにコネクタをインストールする方法を示します。（インストーラーをダウンロードし、インストーラーアイコンを右クリックして、コンテキストメニューから Run As Administrator を選択します。）

Language Selection ポップアップが開いたら、インストール言語を選択し、OK をクリックして Setup ウィンドウに進みます。

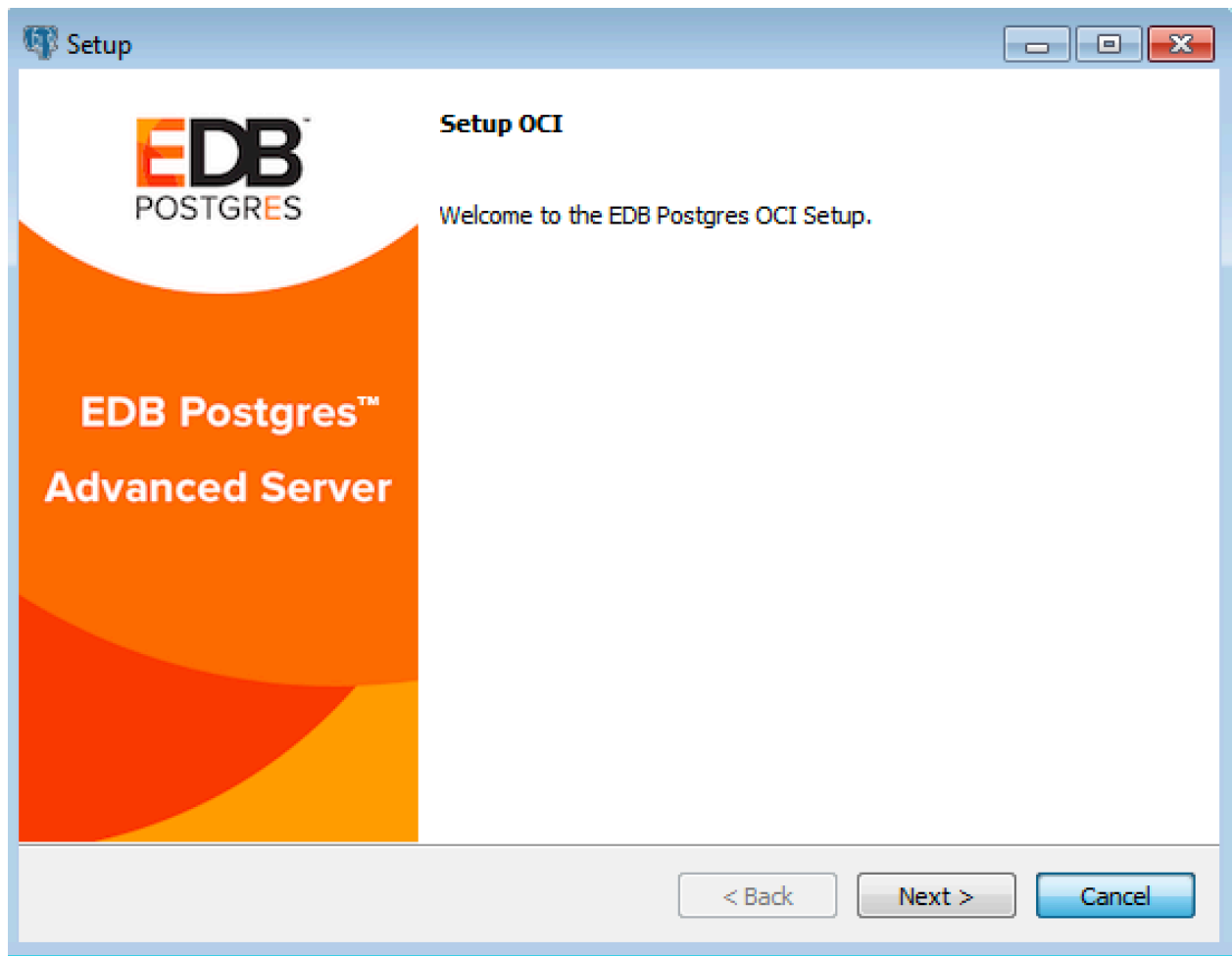


図 2 The OCL Connector Installation wizard

Next をクリックして続行します。

Installation Directory ダイアログを使用してコネクタをインストールするディレクトリを指定し、Next をクリックして続行します。

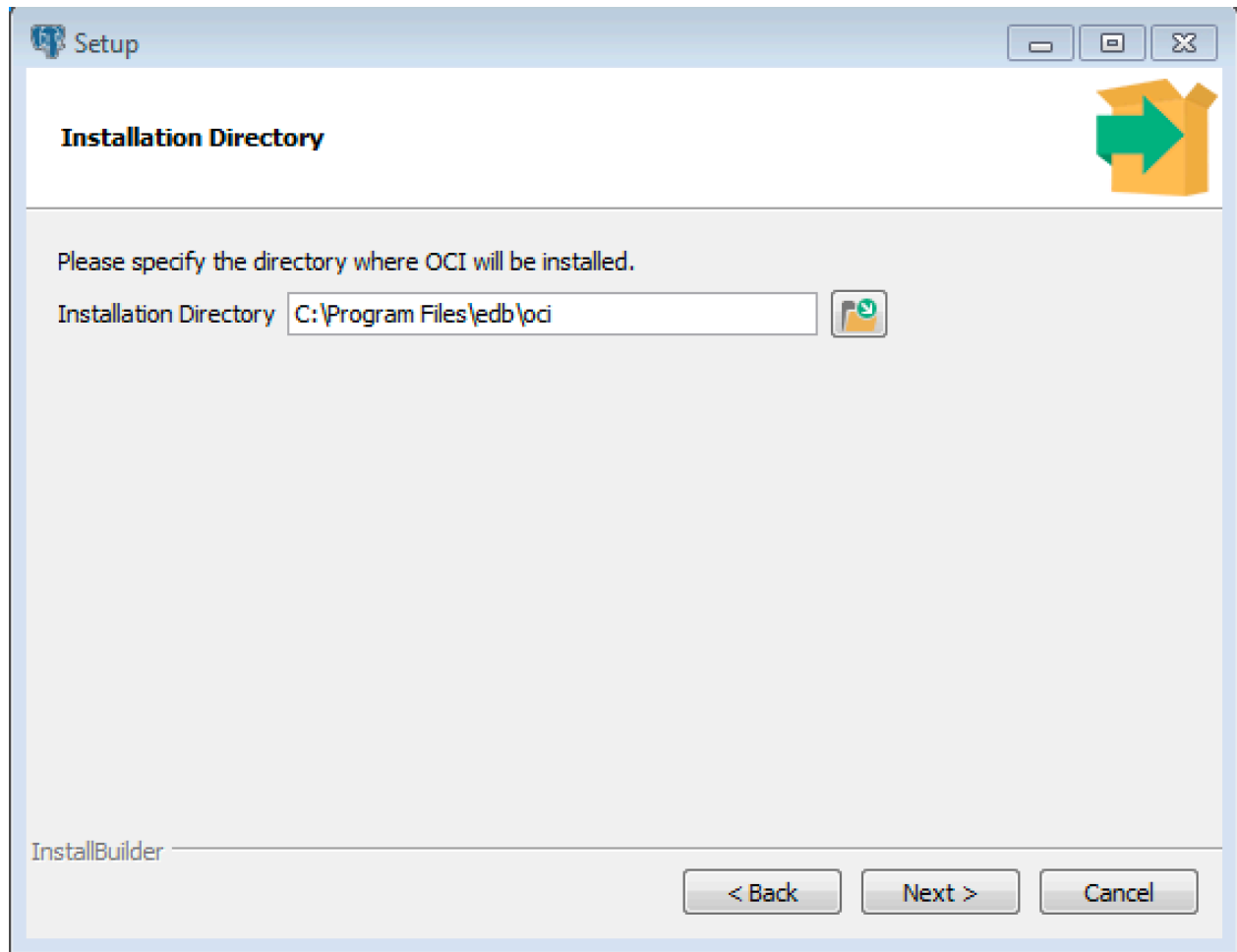


図3 インストールダイアログ

Ready to Install ダイアログで Next をクリックしてインストールを開始します。ポップアップダイアログがインストールウィザードの進行状況を確認します。

ウィザードがセットアップの完了を通知したら、Finish ボタンをクリックしてダイアログを終了します。

StackBuilder Plus を使用して、既存の Advanced Server インストールのコネクタを追加または更新することもできます。StackBuilder Plus を開くには、Windows の Apps メニューまたは Linux の Applications メニューから StackBuilder Plus を選択します。

StackBuilder Plus が開いたら、画面の指示に従います。ツリーコントロールの Database Drivers ノードから EnterpriseDB OCI Connector オプションを選択します。

オンスクリーンウィザードの指示に従って、EnterpriseDB コネクタのインストールを追加または更新します。

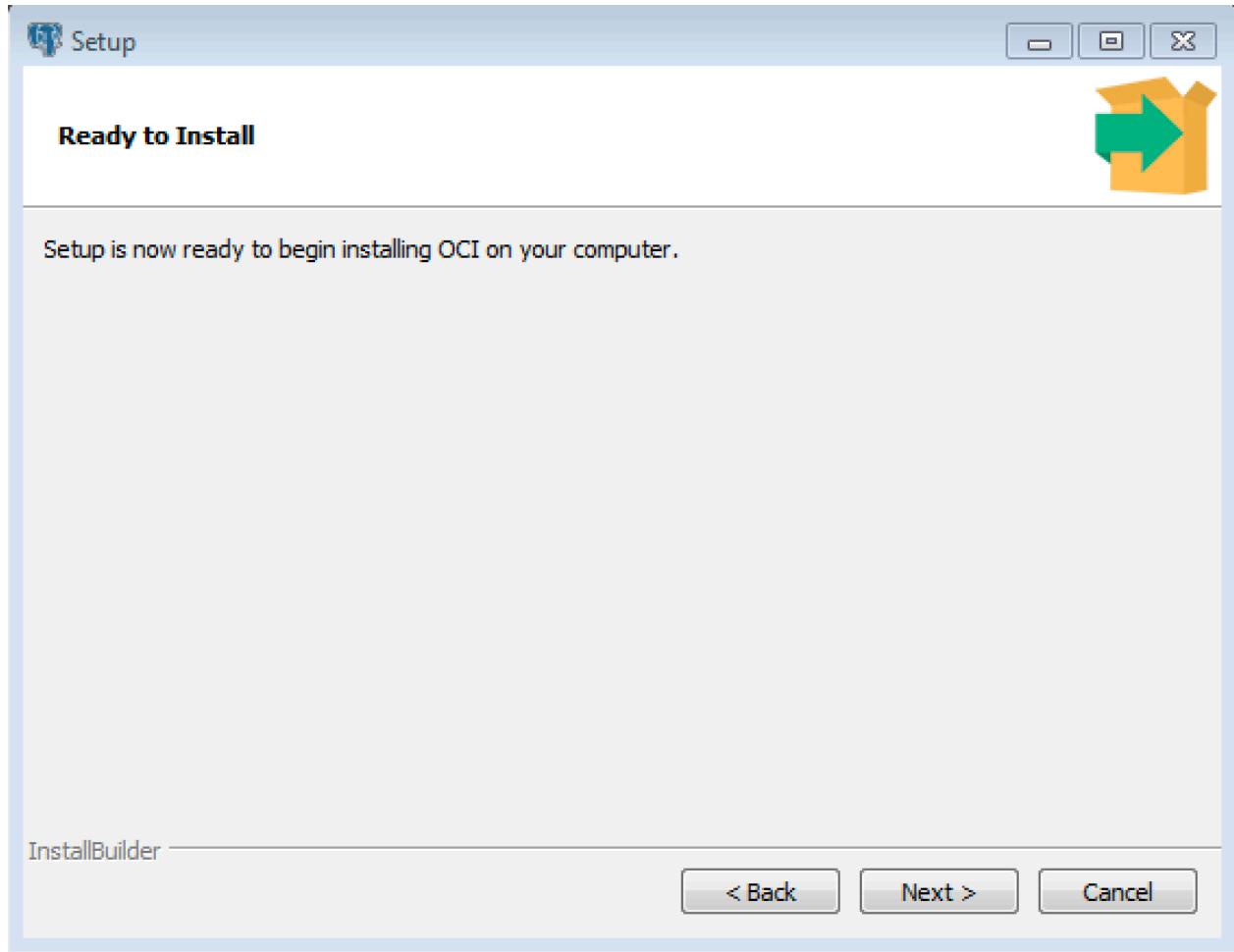


図 4 インストール準備ダイアログ

3.2 接続文字列の作成

OCL コネクタは、Oracle スタイルと Postgres スタイルの両方の接続 URI を受け入れます。接続文字列は、次の Oracle スタイルの形式をとることがあります。

```
[//] [host] [:port] [/dbname]
```

または、次の Postgres スタイルのフォーム:

```
postgres://[user[:password]@] [host] [:port] [/dbname]  
[?param1=value1&...]
```

```
postgresql://[user[:password]@] [host] [:port] [/dbname]  
[?param1=value1&...]
```

Postgres スタイルの URI を使用して、単一の URI で複数のホストコンポーネント（それぞれがオプションのポートコンポーネントを持つ）を指定することもできます。マルチホスト接続文字列の形式は次のとおりです。

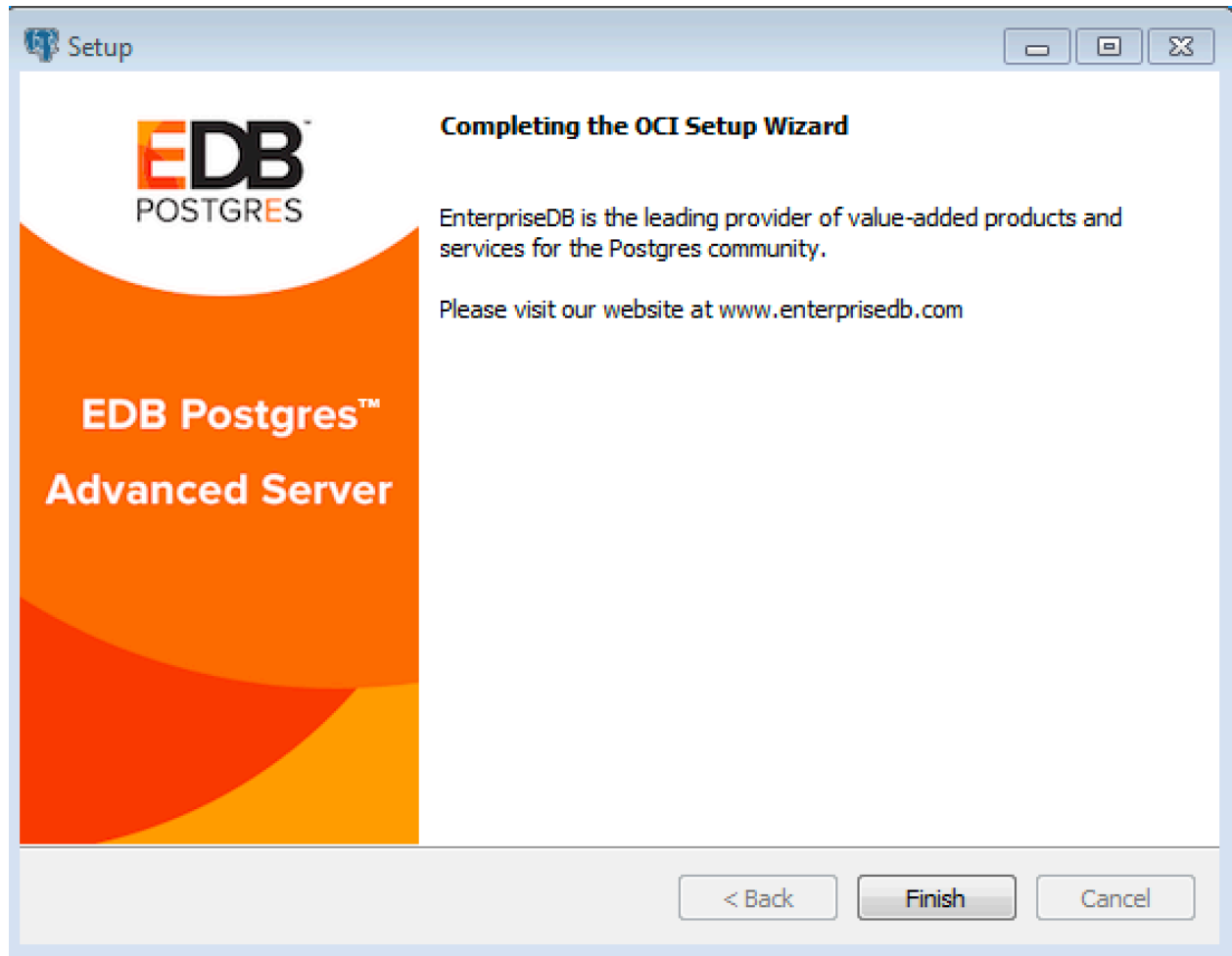


図 5 インストールが完了しました

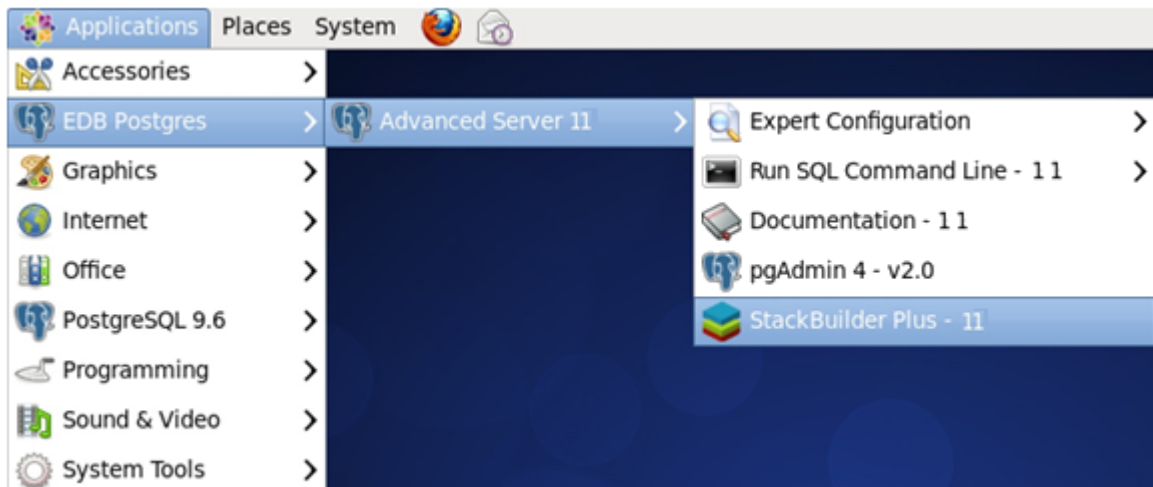


図 6 Starting StackBuilder Plus

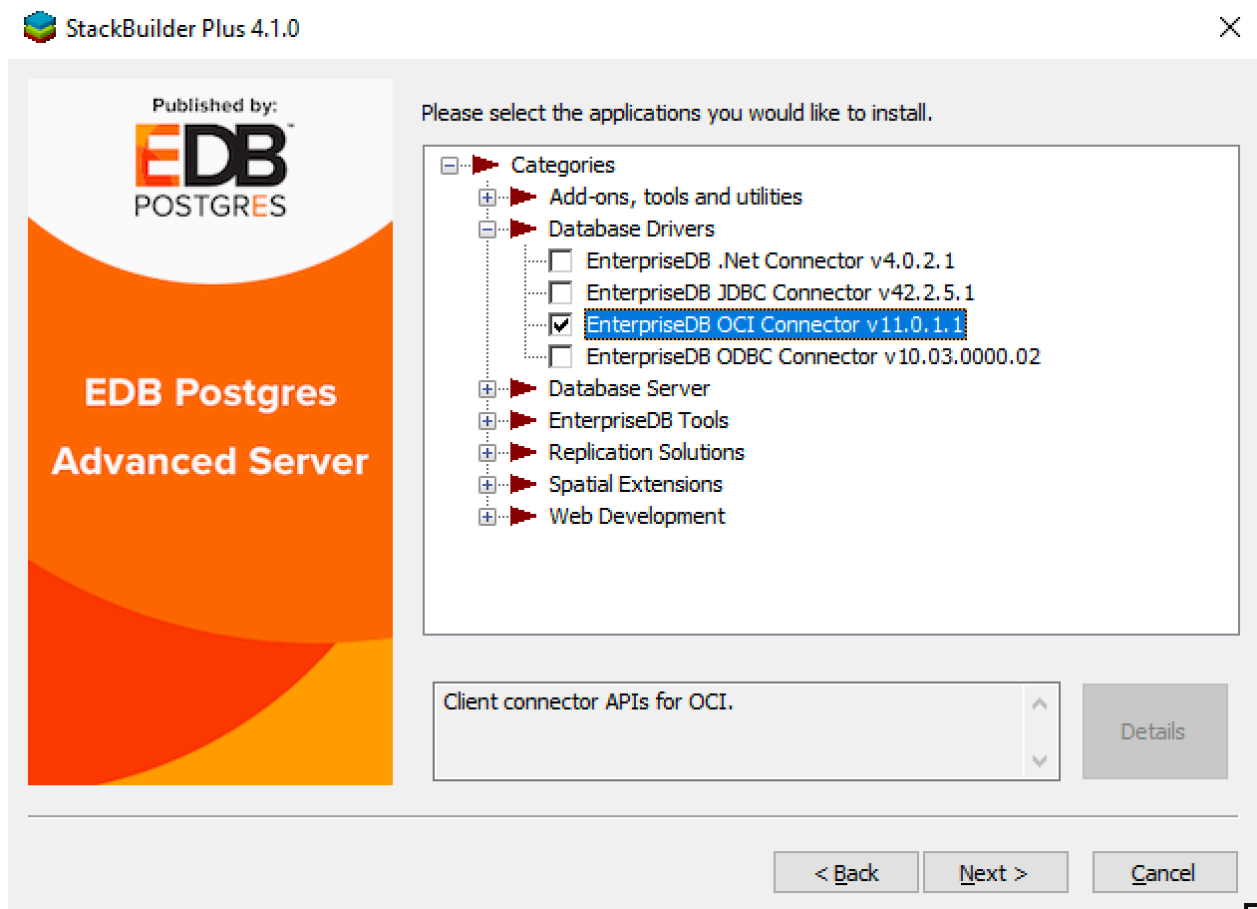


図7 コネクタインストーラーの選択

```
postgresql://host1:port1,host2:port2,host3:port3/
```

どこで:

user は接続しているユーザーの名前です。

password は接続ユーザーに関連付けられたパスワードです。

host は、接続先のホスト名または IP アドレスです。IPV6 アドレスを指定するには、アドレスを角括弧で囲みます。

port は接続先のポート番号です。

dbname は接続しているデータベースの名前です。

paramx=valutex ペアは、追加の（アプリケーション固有の）接続プロパティを指定します。

たとえば、次の各接続文字列は、IP アドレスが 10.0.0.4 のシステムのポート 5444 で edb データベースへの接続を確立します。

```
//10.0.0.4:5444/edb
```

```
postgres://10.0.0.4:5444/edb
```

```
postgresql://10.0.0.4:5444/edb
```

Postgres スタイルの接続文字列の使用の詳細については、“[ここから入手できる PostgreSQL コアドキュメント](https://www.enterprisedb.com/resources/product-documentation)”を参照してください。 <<https://www.enterprisedb.com/resources/product-documentation>>’_。

3.3 プログラムのコンパイルとリンク

EnterpriseDB Open Client Library を使用すると、Oracle Call Interface API を使用して記述されたアプリケーションは、C ソースコードへの最小限の変更で EnterpriseDB データベースに接続し、アクセスできます。EnterpriseDB Open Client Library ファイルの名前は次のとおりです。

Linux の場合:

```
libedboci.so
```

Windows の場合:

```
edboci.dll
```

ファイルは oci/lib サブディレクトリにインストールされます。

サンプルプログラムのコンパイルとリンク

次の例は、Linux 環境でサンプルプログラム edb_demo.c をコンパイルおよびリンクします。edb_demo.c は oci/samples サブディレクトリにあります。

1. ORACLE_HOME および EDB_HOME 環境変数を設定します。

Oracle ホームディレクトリの完全なパス名に ORACLE_HOME を設定します。

例:

```
export ORACLE_HOME=/usr/lib/oracle/xe/app/oracle/product/10.2.0/server
```

ホームディレクトリの完全なパス名に EDB_HOME を設定します。

例:

```
export EDB_HOME=/usr/edb
```

2. LD_LIBRARY_PATH を libpthread.so の完全なパスに設定します。デフォルトでは、libpthread.so は /lib64 にあります。

```
export LD_LIBRARY_PATH=/lib64/lib:$LD_LIBRARY_PATH
```

3. Advanced Server Open Client ライブラリを含めるには、LD_LIBRARY_PATH を設定します。デフォルトでは、libiconv.so.2 は \$EDB_HOME/oci/lib にあります。

```
export
```

```
LD_LIBRARY_PATH=$EDB_HOME/oci:$EDB_HOME/oci/lib:$LD_LIBRARY_PATH
```

4. 次に、OCL API プログラムをコンパイルしてリンクします。

```
cd $EDB_HOME/oci/samples
```

```
make
```

3.4 Ref Cursor Support

Advanced Server Open Client Library は、Oracle と互換性のある PL / SQL プロシージャの OUT パラメーターとして REF CURSOR の使用をサポートしています。サポートは、次の API を通じて提供されます。

- OCIBindByName
- OCIBindByPos
- OCIBindDynamic
- OCISmtPrepare
- OCISmtExecute
- OCISmtFetch
- OCISAttrGet

OCL コネクタは SQLT_RSET データ型もサポートしています。

次の例は、カーソルを開き、出力パラメーターとして REF CURSOR を返すストアードプロシージャを呼び出す方法を示しています。コードサンプルでは、openCursor という名前の PL / SQL プロシージャ（タイプ REF CURSOR のパラメーター OUT）がデータベースサーバー上に作成され、必要なハンドルが割り当てられていると想定しています。

```
char * openCursor =
    "begin \
      openCursor(:cmdRefCursor); \
    end;";
OCISmt *stmtOpenRefCursor;
OCISmt *stmtUseRefCursor;
```

ストアードプロシージャを実行するハンドルを割り当てて、REF CURSOR を開いて使用します:

```
/* Handle for the stored procedure to open the ref cursor */
OCIHandleAlloc((dvoid *) envhp,
               (dvoid **) &stmtOpenRefCursor,
               OCI_HTYPE_STMT,
               0,
               (dvoid **) NULL));
```

```
/* Handle for using the Ref Cursor */
OCIHandleAlloc((dvoid *) envhp,
               (dvoid **) &stmtUseRefCursor,
               OCI_HTYPE_STMT,
               0,
               (dvoid **) NULL));
```

次に、REF CURSOR を開くために使用される PL/SQL ブロックを準備します。

```
OCIStmtPrepare(stmtOpenRefCursor,
               errhp,
               (text *) openCursor,
               (ub4) strlen(openCursor),
               OCI_NTV_SYNTAX,
               OCI_DEFAULT));
```

PL/SQL openCursor OUT パラメータをバインドします。

```
OCIBindByPos(stmtOpenRefCursor,
             &bndplrcl,
             errhp,
             1,
             (dvoid *) &stmtUseRefCursor,
             /* the returned ref cursor */
             0,
             SQLT_RSET,
             /* SQLT_RSET type representing cursor */
             (dvoid *) 0,
             (ub2 *) 0,
             (ub2) 0,
             (ub4) 0,
             (ub4 *) 0,
             OCI_DEFAULT));
```

stmtOpenRefCursor ステートメントハンドルを使用して openCursor プロシージャを呼び出します。

```
OCIStmtExecute(svchp,
               stmtOpenRefCursor,
               errhp,
               1,
               0,
               0,
               0,
               OCI_DEFAULT);
```

この時点で、stmtUseRefCursor ステートメントハンドルにはカーソルへの参照が含まれています。情報を取得するには、参照カーソルの出力変数を定義します。

```
/* Define the output variables for the ref cursor */
OCIDefineByPos(stmtUseRefCursor,
               &defnEmpNo,
               errhp,
               (ub4) 1,
               (dvoid *) &empNo,
               (sb4) sizeof(empNo),
```

(次のページに続く)

(前のページからの続き)

```
SQLT_INT,  
  (dvoid *) 0,  
  (ub2 *) 0,  
  (ub2 *) 0,  
  (ub4) OCI_DEFAULT));
```

次に、結果セットの最初の行をターゲット変数にフェッチします。

```
/* Fetch the cursor data */  
OCIStmtFetch(stmtUseRefCursor,  
             errhp,  
             (ub4) 1,  
             (ub4) OCI_FETCH_NEXT,  
             (ub4) OCI_DEFAULT));
```

3.5 OCL 関数リファレンス

次の表に、OCL コネクタがサポートする機能をリストします。すべてのヘッダーファイルはユーザーが提供する必要があることに注意してください。Advanced Server は、そのようなファイルを提供しません。

3.5.1 関数の接続、承認、初期化

機能	説明
OCIBreak	指定された OCL 機能を中止します。
OCIEnvCreate	OCL 環境を作成します。
OCIEnvInit	OCL 環境ハンドルを初期化します。
OCIInitialize	OCL 環境を初期化します。
OCILogoff	セッションを解放します。
OCILogon	ログオン接続を作成します。
OCILogon2	さまざまなモードでログオンセッションを作成します。
OCIReset	現在の操作/プロトコルをリセットします。
OCIServerAttach	データソースへのアクセスパスを確立します。
OCIServerDetach	データソースへのアクセスを削除します。
OCISessionBegin	ユーザーセッションを作成します。
OCISessionEnd	ユーザーセッションを終了します。
OCISessionGet	セッションプールからセッションを取得します。
OCISessionRelease	セッションを解放します。
OCITerminate	共有メモリサブシステムからデタッチします。

tnsnames.ora ファイルの使用

OCIServerAttach メソッドは、tnsnames.ora ファイルの dblink パラメーターで指定された接続記述子を使用します。Oracle データベースと互換性のある tnsnames.ora ファイルを使用して、データベース接続アドレスを指定します。Advanced Server は、ユーザーのホームディレクトリで tnsnames.ora という名前のファイルを検索します。Advanced Server がユーザーのホームディレクトリに tnsnames.ora ファイルを見つけない場合、TNS_ADMIN で指定されたパスを検索します。

サンプル tnsnames.ora ファイルには以下が含まれます:

```
EDBX =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT = 5444))
  (CONNECT_DATA = (SERVER = DEDICATED) (SID = edb))
)
```

サンプルに含まれていないパラメーターは、Open Client Library によって無視されます。サンプルでは、SID はポート 5444 の localhost で実行されているクラスター内の edb という名前のデータベースを指します。

tnsnames.ora ファイルを使用する OCI Server Attach への C プログラム呼び出しは次のようになります。

```
static text *username = (text *) "enterprisedb";
static text *password = (text *) "edb";
static text *attach_str = "EDBX";
OCI Server Attach(srvhp, errhp, attach_str, strlen(attach_str), 0);
```

tnsnames.ora ファイルがない場合は、接続文字列パラメーターを //localhost:5444/edb の形式で指定します。

3.5.2 ハンドルおよび記述子関数

機能	説明
OCIAttrGet	ハンドル属性を取得します。拡張サーバーは、次のハンドル属性をサポートしています:OCI_ATTR_USERNAME、OCI_ATTR_PASSWORD、OCI_ATTR_SERVER、OCI_ATTR_ENV、OCI_ATTR_SESSION、OCI_ATTR_ROW_COUNT、OCI_ATTR_CHARSET_FORM、OCI_ATTR_CHARSET_ID、EDB_ATTR_STMT_LEVEL_TX、OCI_ATTR_MODE
OCIAttrSet	ハンドル属性を設定します。拡張サーバーは、次のハンドル属性をサポートしています:OCI_ATTR_USERNAME、OCI_ATTR_PASSWORD、OCI_ATTR_SERVER、OCI_ATTR_ENV、OCI_ATTR_SESSION、OCI_ATTR_ROW_COUNT、OCI_ATTR_CHARSET_FORM、OCI_ATTR_CHARSET_ID、EDB_ATTR_STMT_LEVEL_TX、 OCI_ATTR_MODE__
OCIDescriptorAlloc	記述子を割り当てて初期化します。
OCIDescriptorFree	割り当てられた記述子を解放します。
OCIHandleAlloc	ハンドルを割り当てて初期化します。
OCIHandleFree	割り当てられたハンドルを解放します。
OCIParamGet	パラメータ記述子を取得します。
OCIParamSet	パラメータ記述子を設定します。

EDB_ATTR_EMPTY_STRINGS

デフォルトでは、Advanced Server は空の文字列を NULL 値として扱います。EDB_ATTR_EMPTY_STRINGS 環境属性を使用して、空の文字列をマッピングする際の OCL コネクタの動作を制御できます。マッピングの動作を変更するには、OCIAttrSet () 関数を使用して EDB_ATTR_EMPTY_STRINGS を次のいずれかに設定します。

価値	説明
OCI_DEFAULT	空の文字列を NULL 値として扱います。
EDB_EMPTY_STRINGS_NULL	空の文字列を NULL 値として扱います。
EDB_EMPTY_STRINGS_EMPTY	空の文字列を長さゼロの文字列として扱います。

EDB_ATTR_EMPTY_STRINGS の値を見つけるには、OCIAttrGet () をクエリします。

EDB_ATTR_HOLDABLE

Advanced Server は、WITH HOLD カーソルとして実行するステートメントをサポートしています。EDB_ATTR_HOLDABLE 属性は、どのステートメントが WITH HOLD カーソルとして実行されるかを指定します。EDB_ATTR_HOLDABLE 属性は、次の 3 つの値のいずれかに設定できます。

- EDB_WITH_HOLD - WITH HOLD カーソルとして実行
- EDB_WITHOUT_HOLD - プロトコルレベルの準備されたステートメントを使用して実行する
- OCI_DEFAULT - 次の定義を参照

OCIStmt ハンドルまたは OCI_SERVER ハンドルで属性を設定できます。OCI_SERVER ハンドルまたは OCIStmt ハンドルを作成すると、そのハンドルの EDB_ATTR_HOLDABLE 属性が OCI_DEFAULT に設定されます。

OCIAttrSet () を呼び出すことでハンドルの EDB_ATTR_HOLDABLE 属性を変更し、OCIAttrGet () を呼び出すことで属性を取得できます。

Advanced Server は、SELECT ステートメントを実行するときに、OCI_SERVER ハンドルの EDB_ATTR_HOLDABLE 属性を調べます。その属性が EDB_WITH_HOLD に設定されている場合、クエリは WITH HOLD カーソルとして実行されます。

OCI_SERVER ハンドルの EDB_ATTR_HOLDABLE 属性が EDB_WITHOUT_HOLD に設定されている場合、クエリは通常の準備されたステートメントとして実行されます。

OCI_SERVER ハンドルの EDB_ATTR_HOLDABLE 属性が OCI_DEFAULT に設定されている場合、Advanced Server は OCI_SERVER ハンドルの EDB_ATTR_HOLDABLE 属性の値を使用します (OCI_SERVER の 0 “EDB_ATTR_HOLDABLE” に設定され、クエリは ‘8’カーソルとして実行されます。それ以外の場合、クエリはプロトコルで準備されたステートメントとして実行されます)。

EDB_HOLD_CURSOR_ACTION

EDB_HOLD_CURSOR_ACTION 属性は、OCL インターフェイスを使用して WITH HOLD カーソルを作成する方法を変更します。この属性は、次の値のいずれかに設定できます。

- EDB_COMMIT_AFTER_CURSOR - カーソルの作成後にトランザクションをコミットします
- EDB_CURSOR_WITHOUT_XACT_BLK - 新しいトランザクションチェーンを開始しません

- OCI_DEFAULT - 次の定義を参照

次に、属性値について説明します。

OCI_DEFAULT

ステートメントを実行するたびに、OCL はデータベースサーバーのトランザクション状態を調べます。トランザクションがまだ進行中でない場合、OCL は BEGIN ステートメントを実行して新しいトランザクションブロックを作成してから、指定したステートメントを実行します。トランザクションブロックは、OCITransCommit() または OCITransRollback() を呼び出すまで開いたままです。

デフォルトでは、データベースサーバーは、コミットまたはロールバックするときに開いているカーソルをすべて閉じます。あなた（または OCL）が WITH HOLD 句を含むカーソルを宣言した場合、カーソル結果セットはデータベースサーバーに保持され、そのカーソルからフェッチを続けることができます。ただし、トランザクションをロールバックすると、データベースサーバーは開いているカーソルを保持しません。ROLLBACK の後にカーソルからフェッチしようとする、データベースサーバーはエラーを報告します。

EDB_COMMIT_AFTER_CURSOR

アプリケーションがトランザクションをロールバックした後に WITH HOLD カーソルから読み取る必要がある場合、作成する前に EDB_HOLD_CURSOR_ACTION を EDB_COMMIT_AFTER_CURSOR に設定することで、カーソルを作成した直後にトランザクションをコミットするように OCL を調整できますそのようなカーソル。例:

```
ub4          action = EDB_COMMIT_AFTER_CURSOR;

OCIAttrSet(stmt, OCI_HTYPE_STMT, &action, sizeof(action),
    EDB_ATTR_HOLD_CURSOR_ACTION, err);

OCIStmtExecute( ... );
```

EDB_COMMIT_AFTER_CURSOR を使用すると保留中の変更がコミットされることを理解することが重要です。

EDB_CURSOR_WITHOUT_XACT_BLK

アプリケーションが EDB_COMMIT_AFTER_CURSOR によって追加された追加のコミットで適切に実行されない場合、EDB_ATTR_HOLD_CURSOR_ACTION を EDB_CURSOR_WITHOUT_XACT_BLK に設定してみてください。このアクションにより、OCL は新しいトランザクションチェーンを開始しません。トランザクションをコミットまたはロールバックした直後に WITH HOLD カーソルを作成すると、カーソルは独自のトランザクションで作成され、データベースサーバーはそのトランザクションをコミットし、カーソルは持続します。

カーソル宣言がトランザクション内の最初のステートメントではない場合でもエラーが発生する可能性があることを理解することが重要です。カーソルを宣言する前に他のステートメントを実行すると、トランザクションブロックで WITH HOLD カーソルが作成され、エラーが発生した場合（またはアプリケーションが OCITransRollback() を呼び出した場合）にロールバックされる場合があります。

サーバーレベル(OCIserver)または各ステートメントハンドル(OCIstmt)に EDB_HOLD_CURSOR_ACTION を設定できることに注意してください。ステートメント属性が OCI_DEFAULT 以外の値に設定されている場合、

値はステートメントハンドルから取得されます。そうでない場合（ステートメント属性が OCI_DEFAULT に設定されている場合）、値はサーバーハンドルから取得されます。そのため、サーバーハンドルの属性を設定し、ステートメントハンドルの属性を OCI_DEFAULT のままにして、サーバー全体のデフォルトアクションを定義できます。必要に応じて、ステートメントハンドル（またはサーバーハンドル）ごとに異なる値を使用できます。

EDB_ATTR_STMT_LVL_TX

特に指示がない限り、サーバーがエラーを報告するたびに、OCL コネクターは現在のトランザクションを ROLLBACK にします。選択すると、自動 ROLLBACK を `edb_stmt_level_tx` パラメーターでオーバーライドできます。これは、1 つ（または複数）のステートメントがトランザクション内でエラーを発生させた場合でも、トランザクション内の変更を保持します。

OCI Server 属性を `OCIAttrSet()` および `OCIAttrGet()` とともに使用して、`EDB_ATTR_STMT_LEVEL_TX` を有効または無効にできます。デフォルトでは、`edb_stmt_level_tx` は無効になっています。`edb_stmt_level_tx` を有効にするには、クライアントアプリケーションは `OCIAttrSet()` を呼び出す必要があります。

```
OCIServer *server = myServer;
ub1        enabled = 1;

OCIAttrSet(server, OCI_HTYPE_SERVER, &enabled,
            sizeof(enabled), EDB_ATTR_STMT_LEVEL_TX, err);
```

`edb_stmt_level_tx` を無効にするには:

```
OCIServer *server = myServer;
ub1        enabled = 0;

OCIAttrSet(server, OCI_HTYPE_SERVER, &enabled,
            sizeof(enabled), EDB_ATTR_STMT_LEVEL_TX, err);
```

3.5.3 関数のバインド、定義、および説明

機能	説明
OCIBindByName	名前でバインドします。
OCIBindByPos	位置でバインドします。
OCIBindDynamic	バインド後に追加の属性を設定します。
OCIBindArrayOfStruct	一括操作のために構造体の配列をバインドします。
OCIDefineArrayOfStruct	配列の属性を指定します。
OCIDefineByPos	出力変数の関連付けを定義します。
OCIDefineDynamic	定義用の追加属性を設定します。
OCIDescribeAny	既存のスキーマオブジェクトを説明します。
OCIStmtGetBindInfo	バインドおよびインジケータ変数名とハンドルを取得します。
OCIUserCallbackRegister	ユーザー定義のコールバックを定義します。

3.5.4 ステートメント関数

機能	説明
OCIStmtExecute	準備された SQL ステートメントを実行します。
OCIStmtFetch	データの行をフェッチします（非推奨）。
OCIStmtFetch2	データの行をフェッチします。
OCIStmtPrepare	SQL ステートメントを準備します。
OCIStmtPrepare2	SQL ステートメントを準備します。
OCIStmtRelease	ステートメントハンドルを解放します。

3.5.5 トランザクション関数

機能	説明
OCITransCommit	トランザクションをコミットします。
OCITransRollback	トランザクションをロールバックします。

3.5.6 XA 関数

機能	説明
xaoEnv	OCL 環境ハンドルを返します。
xaoSvcCtx	OCL サービスコンテキストを返します。

xaoSvcCtx

xaoSvcCtx 関数を使用するには、xaoSvcCtx または xa_open 接続文字列形式の拡張機能を次のように提供する必要があります。

```
Oracle_XA{+<required_fields> ...}
```

ここで、required_fields は次のとおりです。

HostName=host_ip_address は Advanced Server データベースの IP アドレスを指定します。

PortNumber=host_port_number は、Advanced Server が実行されているポート番号を指定します。

SqlNet=dbname はデータベース名を指定します。

Acc=P/username/password はデータベースのユーザー名とパスワードを指定します。* password *は省略できます。その場合、フィールドは Acc=P/username/ として指定されます。

AppName=app_id は、アプリケーションを識別する番号を指定します。

以下は、接続文字列の例です。

```
Oracle_XA+HostName=192.168.1.1+PortNumber=1533+SqlNet=XE+Acc=P/user/  
↪password+AppName=1234
```

3.5.7 日付および日時関数

機能	説明
OCIDateAddDays	日数を加算または減算します。
OCIDateAddMonths	月数を加算または減算します。
OCIDateAssign	日付を割り当てます。
OCIDateCheck	指定された日付が有効かどうかを確認してください。
OCIDateCompare	2 つの日付を比較します。
OCIDateDaysBetween	2 つの日付間の日数を見つけます。
OCIDateFromText	文字列を日付に変換します。
OCIDateGetDate	日付の日付部分を取得します。
OCIDateGetTime	日付の時刻部分を取得します。
OCIDateLastDay	月の最終日の日付を取得します。
OCIDateNextDay	翌日の日付を取得します。
OCIDateSetDate	日付の日付部分を設定します。
OCIDateSetTime	日付の時刻部分を設定します。
OCIDateSysDate	現在のシステムの日付と時刻を取得します。
OCIDateToText	日付を文字列に変換します。
OCIDateTimeAssign	日時の割り当てを実行します。
OCIDateTimeCheck	日付が有効かどうかを確認してください。
OCIDateTimeCompare	2 つの日時値を比較します。
OCIDateTimeConstruct	日時記述子を作成します。
OCIDateTimeConvert	ある日時タイプを別の日時タイプに変換します。
OCIDateTimeFromArray	サイズ OCI_DT_ARRAYLEN の配列を OCIDateTime 記述子に変換します。
OCIDateTimeFromText	指定された形式に従って、指定された文字列を OCIDateTime 記述子の Oracle 日時型に変換します。
OCIDateTimeGetDate	日時値の日付部分を取得します。
OCIDateTimeGetTime	日時値の時刻部分を取得します。
OCIDateTimeGetTimeZoneName	日時値のタイムゾーン名部分を取得します。
OCIDateTimeGetTimeZoneOffset	日時値のタイムゾーン（時間、分）部分を取得します。
OCIDateTimeSubtract	2 つの日時値を入力として受け取り、それらの差を間隔として返します。
OCIDateTimeSysTimeStamp	タイムゾーン付きのタイムスタンプとしてシステムの現在の日付と時刻を取得します。
OCIDateTimeToArray	OCIDateTime 記述子を配列に変換します。
OCIDateTimeToText	指定された形式に従って、指定された日付を文字列に変換します。

3.5.8 インターバル関数

機能	説明
OCIIntervalAdd	2つの間隔値を追加します。
OCIIntervalAssign	ある間隔値を別の間隔値にコピーします。
OCIIntervalCompare	2つの間隔値を比較します。
OCIIntervalGetDaySecond	間隔から日、時間、分、秒、および小数秒を抽出します。
OCIIntervalSetDaySecond	間隔内の日、時間、分、秒、および小数秒を変更します。
OCIIntervalGetYearMonth	間隔から年と月の値を抽出します。
OCIIntervalSetYearMonth	間隔で年と月の値を変更します。
OCIIntervalDivide	OCINumber 値による OCIInterval 値の除算を実装します。
OCIIntervalMultiply	OCIInterval 値と OCINumber 値の乗算を実装します。
OCIIntervalSubtract	あるインターバル値を別のインターバル値から減算します。
OCIIntervalToText	間隔から文字列を外挿します。
OCIIntervalCheck	間隔値の妥当性を検証します。
OCIIntervalToNumber	OCIInterval 値を OCINumber 値に変換します。
OCIIntervalFromNumber	OCINumber 値を OCIInterval 値に変換します。
OCIDateTimeIntervalAdd	OCIInterval 値を OCIDatetime 値に追加し、結果として OCIDatetime 値を生成します。
OCIDateTimeIntervalSub	OCIDatetime 値から OCIInterval 値を減算し、OCIDatetime 値を生成します。
OCIIntervalFromText	テキスト文字列を間隔に変換します。
OCIIntervalFromTZ	時間帯の指定を間隔値に変換します。

3.5.9 数字関数

機能	説明
OCINumberAbs	絶対値を計算します。
OCINumberAdd	NUMBER を追加します。
OCINumberArcCos	逆余弦を計算します。
OCINumberArcSin	逆正弦を計算します。
OCINumberArcTan	逆正接を計算します。
OCINumberArcTan2	2 つの NUMBER の逆正接を計算します。
OCINumberAssign	1 つの NUMBER を別の NUMBER に割り当てます。
OCINumberCeil	NUMBER の上限を計算します。
OCINumberCmp	NUMBER を比較します。
OCINumberCos	コサインを計算します。
OCINumberDec	NUMBER をデクリメントします。
OCINumberDiv	2 つの NUMBER を分割します。
OCINumberExp	e を指定された NUMBER 乗します。
OCINumberFloor	NUMBER のフロアを計算します。
OCINumberFromInt	整数を Oracle NUMBER に変換します。
OCINumberFromReal	実数を Oracle NUMBER に変換します。
OCINumberFromText	文字列を Oracle NUMBER に変換します。
OCINumberHypCos	双曲線余弦を計算します。
OCINumberHypSin	双曲線正弦を計算します。
OCINumberHypTan	双曲線正接を計算します。
OCINumberInc	NUMBER をインクリメントします。
OCINumberIntPower	指定された基数を整数の累乗にします。
OCINumberIsInt	NUMBER が整数かどうかをテストします。
OCINumberIsZero	NUMBER がゼロかどうかをテストします。
OCINumberLn	自然対数を計算します。
OCINumberLog	任意の底の対数を計算します。
OCINumberMod	モジュロ除算。
OCINumberMul	NUMBER を掛けます。
OCINumberNeg	NUMBER を無効にします。
OCINumberPower	基数 e へのべき乗。
OCINumberPrec	NUMBER を指定された小数点以下の桁数に丸めます。
OCINumberRound	NUMBER を指定された小数位に丸めます。
OCINumberSetPi	NUMBER を Pi に初期化します。
OCINumberSetZero	NUMBER をゼロに初期化します。
OCINumberShift	指定された小数点以下の桁数をシフトして、10 を掛けます。

次のページに続く

表 2 – 前のページからの続き

OCINumberSign	NUMBER の符号を取得します。
OCINumberSin	サインを計算します。
OCINumberSqrt	NUMBER の平方根を計算します。
OCINumberSub	NUMBER を引きます。
OCINumberTan	タンジェントを計算します。
OCINumberToInt	NUMBER を整数に変換します。
OCINumberToReal	NUMBER を実数に変換します。
OCINumberToRealArray	NUMBER の配列を実際の配列に変換します。
OCINumberToText	NUMBER を文字列に変換します。
OCINumberTrunc	指定された小数位で NUMBER を切り捨てます。

3.5.10 文字列関数

機能	説明
OCIStrAllocSize	割り当てられた文字列メモリのサイズをバイト単位で取得します。
OCIStrAssign	文字列に文字列を割り当てます。
OCIStrAssignText	文字列にテキスト文字列を割り当てます。
OCIStrPtr	文字列ポインタを取得します。
OCIStrResize	文字列メモリのサイズを変更します。
OCIStrSize	文字列サイズを取得します。

3.5.11 カートリッジサービスとファイル I/O インターフェイス関数

機能	説明
OCIFileClose	開いているファイルを閉じます。
OCIFileExists	ファイルが存在するかどうかを確認してください。
OCIFileFlush	バッファリングされたデータをファイルに書き込みます。
OCIFileGetLength	ファイルの長さを取得します。
OCIFileInit	OCIFile パッケージを初期化します。
OCIFileOpen	ファイルを開きます。
OCIFileRead	ファイルからバッファに読み込みます。
OCIFileSeek	ファイル内の現在の位置を変更します。
OCIFileTerm	OCIFile パッケージを終了します。
OCIFileWrite	ファイルに buflen バイトを書き込みます。

3.5.12 LOB 関数

機能	説明
OCILobRead	LOB 値（または LOB 値の一部）を返します。
OCILOBWriteAppend	LOB 値にデータを追加します。
OCILobGetLength	LOB 値の長さを返します。
OCILobTrim	LOB 値の末尾からデータを切り取ります。
OCILobOpen	他の LOB 関数で使用するために LOB 値を開きます。
OCILobClose	LOB 値を閉じます。

3.5.13 その他の機能

機能	説明
OCIClientVersion	クライアントライブラリのバージョンを返します。
OCLErrorGet	エラーメッセージを返します。
OCIPGErrorGet	libpq またはサーバーによって報告されたネイティブエラーメッセージを返します。署名は次のとおりです。 剣 OCIPGErrorGet (dvoid / * hndlp、ub4 recordno、OraText / * errcodep、ub4 errbufsiz、OraText / * bufp、ub4 bufsiz、ub4 タイプ)
OCIPasswordChange	パスワードを変更します。
OCIPing	接続とサーバーがアクティブであることを確認します。
OCIServerVersion	Oracle バージョン文字列を取得します。

3.5.14 サポートされているデータ型

機能	説明
ANSI_DATE	ANSI 日付
SQLT_AFC	ANSI 固定文字
SQLT_AVC	ANSI 可変文字
SQLT_BDOUBLE	バイナリダブル
SQLT_BIN	バイナリデータ
SQLT_BFLOAT	バイナリフロート
SQLT_CHR	文字列
SQLT_DAT	Oracle の日付
SQLT_DATE	ANSI 日付
SQLT_FLT	フロート
SQLT_INT	整数
SQLT_LBI	ロングバイナリ
SQLT_LNG	長い
SQLT_LVB	長いロングバイナリ
SQLT_LVC	ロングロング (文字)
SQLT_NUM	Oracle 数値
SQLT_ODT	OCL 日付タイプ
SQLT_STR	ゼロ終了文字列
SQLT_TIMESTAMP	タイムスタンプ
SQLT_TIMESTAMP_TZ	タイムゾーン付きのタイムスタンプ
SQLT_TIMESTAMP_LTZ	ローカルタイムゾーンのタイムスタンプ
SQLT_UIN	符号なし整数
SQLT_VBI	VCS 形式のバイナリ
SQLT_VCS	可変文字
SQLT_VNU	先行する長さのバイトを持つ番号
SQLT_VST	OCL 文字列タイプ

3.6 OCL エラーコード-リファレンス

次の表に、OCL コネクタによって定義されたエラーコードマッピングを示します。データベースサーバーがエラーコードまたは状態（1 列目または 2 列目に表示）を報告すると、OCL は値を 3 列目に表示される互換性のある値に変換します。

エラーコード	条件名	Oracle エラーコード
42601	構文エラー	ORA-16945
42P01	undefined_table	ORA-00942
02000	no_data	ORA-01403
08000	connection_exception	ORA-12545
08003	connection_does_not_exist	ORA-12545
08006	connection_failure	ORA-12545
08001	sqlclient_unable_to_establish_sqlconnection	ORA-12545
08004	sqlserver_rejected_establishment_of_sqlconnection	ORA-12545
25000	invalid_transaction_state	ORA-01453
08007	transaction_resolution_unknown	ORA-01453
0A000	feature_not_supported	ORA-03001
22012	division_by_zero	ORA-01476
2200B	escape_character_conflict	ORA-01424
22019	invalid_escape_character	ORA-00911
2200D	invalid_escape_octet	ORA-01424
22025	invalid_escape_sequence	ORA-01424
22P06	nonstandard_use_of_escape_character	ORA-01424
2200C	invalid_use_of_escape_character	ORA-01424
22004	null_value_not_allowed	ORA-01400
23000	integrity_constraint_violation	ORA-00001
23505	unique_violation	ORA-00001
40P01	t_r_deadlock_detected	ORA-00060
42701	duplicate_column	ORA-01430
53000	リソースが不足しています	ORA-01659
53100	disk_full	ORA-01659
53200	out_of_memory	ORA-82100
42P07	duplicate_table	ORA-00955
21000	cardinality_violation	ORA-01427
22003	numeric_value_out_of_range	ORA-01426
22P02	invalid_text_representation	ORA-01858
28000	invalid_authorization_specification	ORA-01017

次のページに続く

表 3 – 前のページからの続き

エラーコード	条件名	Oracle エラーコード
28P01	invalid_password	ORA-01017
2200F	zero_length_character_string	ORA-01425
42704	undefined_object	ORA-01418
2BP01	dependent_objects_still_exist	ORA-02429
22027	trim_error	ORA-30001
22001	string_data_right_truncation	ORA-01401
22002	null_value_no_indicator_parameter	ORA-01405
22008	datetime_field_overflow	ORA-01800
44000	with_check_option_violation	ORA-01402
01007	warning_privilege_not_granted	ORA-00000
01006	warning_privilege_not_revoked	ORA-00000
02001	no_additional_dynamic_result_sets_returned	ORA-00000
03000	sql_statement_not_yet_complete	ORA-00000
08P01	protocol_violation	ORA-00000
23001	restrict_violation	ORA-00000
23502	not_null_violation	ORA-00000
23505	foreign_key_violation	ORA-00000
23514	check_violation	ORA-00000
24000	invalid_cursor_state	ORA-01001
26000	invalid_sql_statement_name	ORA-00000
42830	invalid_foreign_key	ORA-00000
55006	object_in_use	ORA-00000
55P03	lock_not_available	ORA-00054
72000	snapshot_too_old	ORA-01555

Postgres エラーコードの詳細については、PostgreSQL コアの“ドキュメント”を参照してください。 <<https://www.postgresql.org/docs/12/static/errcodes-appendix.html>>‘_。

3.7 マルチスレッドのサポート

OCL はマルチスレッド環境でサポートされています。mode パラメーターの値として OCI_THREADED を指定して OCIEnvNlsCreate() を呼び出すことにより、マルチスレッド環境でマルチスレッドを有効化/使用できます。

```
retCode = OCIEnvNlsCreate( &envp,  
                           OCI_THREADED,  
                           NULL,  
                           NULL,  
                           NULL,  
                           NULL,  
                           0,  
                           NULL,  
                           0,  
                           0 );
```

後続の OCIEnvNlsCreate() への呼び出しもすべて OCI_THREADED で行わなければなりません。

マルチスレッドアプリケーションがスレッドセーフなオペレーティングシステムで実行されている場合、OCI ライブラリは、各環境ハンドルのアプリケーションのミューテックスを管理します。

第 4 章

OCL トレースの生成

OCL トレースオプションは、指定された `OCI_DEBUG_LOG file 1 OCI_DEBUG_LOG file 2`）を追加すると、トレースファイルが特定の場所に生成されます。

トレースファイルは、接続ごとにテキストファイル（読み取り可能）形式で生成されます。

注釈: OCL トレースはデフォルトで無効になっています。

OCL トレースを生成するには:

1. OCL の EDB クライアント側のトレースを有効にします。以下の環境変数を設定することにより、OCL トレースを有効にできます。

```
export OCI_DEBUG_LEVEL=4
```

```
export OCI_DEBUG_LOG=oci_log_file
```

2. 環境変数をエクスポートしたら、アプリケーションを実行します。OCL トレースファイルは、指定されたディレクトリに生成されます。

第 5 章

SSL の使用

EDB Postgres Advanced Server は、SSL 接続を使用してクライアント/サーバー通信を暗号化し、セキュリティを強化するネイティブサポートを提供します。OCL では、`sslmode` パラメーターを `verify-full` または `verify-ca` に設定し、検証するルート証明書をシステムに提供することで制御されます。

SSL 設定の手順:

1. サーバーおよびクライアント側の証明書を構成します。SSL クライアントおよびサーバー側の証明書の構成に関する詳細については、PostgreSQL SSL の“ドキュメント”を参照してください。 <<https://www.postgresql.org/docs/12/libpq-ssl.html>>‘_。
2. SSL OCL 接続を有効にします。

OCL クライアントアプリケーションでは、Session の `EDB_ATTR_SSL` 属性を設定することで SSL モードを有効にできます。

```
char*sslmode= "verify-full";
retValue=OCIAttrSet((dvoid*)authp, (ub4)OCI_HTYPE_SESSION,
                    (dvoid*)sslmode, (ub4)strlen((char*)sslmode),
                    (ub4)EDB_ATTR_SSL, errhp);
```

注釈: インストールディレクトリで利用可能な `edboci.h` ヘッダーファイルに `EDB_ATTR_SSL` が定義されています。

3. SSL 属性を設定した後、`OCILogon` 関数を使用して接続を作成できます。

```
OCILogon(pEnv, pError, &pSvc, (OraText*)pUsername, ub4)UsernameLen,
        (OraText*)pPassword, (ub4)PasswordLen,
        (OraText*)pDatabase, (ub4)DatabaseLen);
```

サーバーが認証されると、クライアントは機密データを渡す準備が整います。

サポートされている SSL モードオプションの詳細については、以下を参照してください。

<https://www.postgresql.org/docs/12/libpq-ssl.html#LIBPQ-SSL-SSLMODE-STATEMENTS>

第 6 章

結論

EDB Postgres OCL ガイド

Copyright©2007-2019 EnterpriseDB Corporation。

無断複写・転載を禁じます。

EnterpriseDB®Corporation

34 Crosby Drive, Suite 201, Bedford, MA 01730, アメリカ

T +1 781 357 3390 F +1 978 467 1307 E

info@enterprisedb.com

www.enterprisedb.com

- EDB は、存在する場合、EDB 製品のログオン UI の設計、コーディングのベストプラクティスの確立、レビュー、および入力検証の検証を行います。EDB は追加の入力コンポーネントに対して同じアプローチに従いますが、製品の性質上、限定された検証が可能な信頼できるユーザーによって入力および送信されるフリーフォーム SQL、WMI またはその他の文字列を受け入れる必要がある場合があります。このような場合、ユーザーが誤った入力やその他の危険な入力を防ぐことはできません。
- EDB は、将来、認証された信頼できるユーザーからのフリーフォーム SQL、WMI またはその他の潜在的に危険な入力を受け入れる製品に機能を追加する権利を留保しますが、そのような機能はすべて、可能な限り最小のリスクを提供するように設計およびテストされます可能であれば、スーパーユーザーまたは同等の特権が必要です。
- EDB は、すべての潜在的な脅威を予測できることを保証しません。したがって、すべての潜在的な脆弱性が対処または検討されたことをプロセスが完全に保証することはできません。

索引

- Authorize Functions, 23
- Bind Functions, 29
- Cartridge Services and File I/O Interface Functions, 35
 - compiling and linking program, 19
- Connect Functions, 23
- Date and Datetime Functions, 31
- Define Functions, 29
- Describe Functions, 29
- EDB_ATTR_HOLDABLE, 26
- EDB_ATTR_STMT_LVL_TX, 28
- EDB_HOLD_CURSOR_ACTION, 26
- graphical installation, 13
- Handle and Descriptor Functions, 25
- Initialize Functions, 23
- Installation on Debian, 12
- Installation on SLES 12, 10
- Installation on Ubuntu, 12
- Interval Functions, 32
- LOB Functions, 35
- Miscellaneous Functions, 36
- Number Functions, 33
- OCL エラーコード-リファレンス, 38
- OCL コネクタのインストールと設定, 8
- OCL トレースの生成, 41
- OCL 関数リファレンス, 23
- Open Client Library, 7
- Ref Cursor Support, 20
- Ref cursor support, 20
- SSL の使用, 43
- Statement Functions, 29
- String Functions, 35
- Supported Data Types, 37
- Supported server versions, 5
- tnsnames.ora, 23
- Transaction Functions, 29
- XA Functions, 30
- xaoSvcCtx, 30
- サポートされているプラットフォーム, 5
- プログラムのコンパイルとリンク, 19
- マルチスレッドのサポート, 40
- 結論, 45
- 新機能, 3
- 接続文字列の作成, 15