Oracle® Database
データベース管理者リファレンス
19c for Linux and UNIX System-Based Operating Systems

F16143-04(原本部品番号:E96356-05)

2023年6月

タイトルおよび著作権情報

Oracle Databaseデータベース管理者リファレンス, 19c for Linux and UNIX-Based Operating Systems

F16143-04

原本著者: Binika Kumar

原本協力著者:Subhash Chandra、Prakash Jashnani

原本協力者: Kevin Flood、Pat Huey、Clara Jaeckel、Emily Murphy、Terri Winters、Subhranshu Banerjee、Mark Bauer、Robert Chang、Jonathan Creighton、Sudip Datta、Thirumaleshwara Hasandka、Joel Kallman、George Kotsovolos、Richard Long、Rolly Lv、Padmanabhan Manavazhi、Matthew Mckerley、Krishna Mohan、Rajendra Pingte、Hanlin Qian、Janelle Simmons、Roy Swonger、Michael Coulter、Robert Achacoso、Malai Stalin、Ramesh Chakravarthula、David Price、Douglas Williams、Joseph Therrattil Koonen、Binoy Sukumaran、and Sumanta Chatterjee。

目次

はじめに

このマニュアルでは、次の各プラットフォームでOracle Database 19cを管理および構成する方法について、プラットフォームごとに説明します。

- Oracle Solaris
- Linux
- IBM AIX on POWER Systems (64ビット)
- HP-UX Itanium

このマニュアルでは、『Oracle Database管理者ガイド』を補足的に説明します。

対象読者

このマニュアルは、Oracle Database 19cの管理および構成を担当する方を対象としています。Oracle RACを構成する場合は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

ドキュメントのアクセシビリティ

Oracleのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWebサイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracleサポートへのアクセス

サポートを購入したオラクル社のお客様は、My Oracle Supportを介して電子的なサポートにアクセスできます。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info)か、聴覚に障害のあるお客様は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

ダイバーシティとインクルージョン

Oracleはダイバーシティ&インクルージョンに積極的に取り組んでいます。Oracleは、ソート・リーダーシップと革新性を高める社員の多様性を尊重し、その価値を重んじています。従業員、お客様、パートナー様にポジティブな影響をもたらすインクルーシブな文化を醸成する私たちのイニシアティブの一環として、製品やドキュメントからインセンシティブな用語を取り除くように努めています。 また、Oracle製品および業界標準が進化する中、お客様の既存の技術との互換性を維持する必要性およびサービスの継続性確保の要求にも留意しています。このような技術的な制限により、当社のインセンシティブな用語を削除する取組みは継続中であり、時間と皆様のご協力が必要となります。

Java Accessibilityを実装するためのJava Access Bridgeの設定

Microsoft Windowsシステムでアシスティブ・テクノロジがJava Accessibility APIを使用できるようにJava Access Bridgeをインストールします。

Java Access Bridgeは、Java Accessibility APIを実装するJavaアプリケーションおよびアプレットをMicrosoft Windowsシステム上のユーザー補助テクノロジから可視にするためのテクノロジです。

Java Access Bridgeの使用に必要なアシスティブ・テクノロジのサポートされる最小バージョンの詳細は、『*Java Platform, Standard Edition Java*アクセシビリティ・ガイド』を参照してください。また、インストール手順とテスト手順およびJava

Access Bridgeの使用方法の詳細は、このガイドを参照してください。

関連項目

- Java Platform, Standard Edition Javaアクセシビリティ・ガイド

関連ドキュメント

重要な情報は、Oracle Databaseドキュメント・ライブラリにあるプラットフォーム固有のリリース・ノート、インストレーション・ガイドおよびExamplesインストレーション・ガイドを参照してください。

表記規則

このドキュメントでは次の表記規則を使用します。

表記規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。
イタリック体	イタリックは、ユーザーが特定の値を指定するプレースホルダー変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

コマンド構文

UNIXのコマンド構文は、固定幅フォントで表示されます。ドル記号($)、シャープ記号(#)、パーセント記号(%)はUNIXのコマンド・プロンプトです。これらの記号をコマンドの一部として入力しないでください。このマニュアルでは、コマンド構文に次の表記規則を使用しています。

表記規則	説明

バックスラッシュ ¥	バックスラッシュは、UNIX コマンドの行の継続を表す記号です。コマンド例が 1 行に入りきらない場合に使用します。コマンドは、表示どおりにバックスラッシュを付けて入力するか、またはバックスラッシュなしで 1 行に入力します。
	
	`dd if=/dev/rdsk/c0t1d0s6 of=/dev/rst0 bs=10b ¥`
	`count=10000`

中カッコ{ }	中カッコは、必須の入力項目を表します。 `.DEFINE {macro1}`
大カッコ [ ]	大カッコは、カッコ内の項目を任意に選択することを表します。 `cvtcrt termname [outfile]`

表記規則	説明
省略記号...	省略記号は、同じ項目を任意の数だけ繰り返すことを表します。 `CHKVAL fieldname value1 value2 ... valueN`
イタリック体	イタリック体は、変数を表します。変数には値を代入します。 `library_name`
縦線 |	縦線は、大カッコまたは中カッコ内の複数の選択項目の区切りに使用します。 `FILE filesize [K|M]`

用語

このマニュアルではUNIXオペレーティング・システムの名前を短縮して使用しています。次のとおりです。

オペレーティング・システム	短縮名
• Oracle Solaris on SPARC(64-Bit) • Oracle Solaris on x86-64(64-Bit)	Oracle Solaris ノート: 特定のアーキテクチャにおける Oracle Solaris の情報の違いについては、本文中に記載されています。
• Linux x86-64	Linux ノート: 特定のアーキテクチャにおける Linux の情報の違いについては、本文中に記載されています。

ドキュメントへのアクセス

このリリースには、プラットフォーム固有のドキュメントと汎用の製品ドキュメントがあります。プラットフォーム固有のドキュメントには、Oracle製品を特定のプラットフォームにインストール、構成および使用する方法が記載されています。ドキュメントには、Adobe Portable Document Format(PDF)形式とHTML形式があります。

次のURLからすべてのOracleドキュメントにアクセスできます。

http://docs.oracle.com/en/

ノート:

プラットフォーム固有のドキュメントは、リリース時点のものです。Oracle Technology Network の Web サイ

トから最新の情報を入手することをお薦めします。

1 Oracle Databaseの管理

この章では、UNIXベースのオペレーティング・システムでOracle Databaseを管理する方法について説明します。内容は次のとおりです。

- 概要
- 環境変数
- 初期化パラメータ
- 診断データの管理

関連項目:

- Oracle Databaseの管理に関するプラットフォーム固有の情報は、このマニュアルの付録を参照してください。
- Oracle Databaseの管理に関する一般的な情報は、『Oracle Database管理者ガイド』および『2日でデータベース管理者』を参照してください。

1.1 概要

Oracle Databaseを使用するには、Oracle Databaseの環境変数、パラメータおよびユーザー設定を設定する必要があります。この章では、Oracle Databaseの各種設定について説明します。

Oracle Databaseのファイルおよびプログラムでは、疑問符(?)は環境変数ORACLE_HOMEの値を表します。たとえば、Oracle Databaseでは、次のSQL文中の疑問符は、Oracleホーム・ディレクトリのフルパス名に展開されます。

```
SQL> ALTER TABLESPACE TEMP ADD DATAFILE '?/dbs/temp02.dbf' SIZE 200M
```

同様に、アットマーク(@)記号は環境変数ORACLE_SIDを表します。たとえば、ファイルが現行のインスタンスに属していることを示す場合は、次のコマンドを実行します。

```
SQL> ALTER TABLESPACE tablespace_name ADD DATAFILE tempfile@.dbf
```

1.2 環境変数

この項では、通常使用されるOracle DatabaseおよびオペレーティングシステムのOracle Databaseの環境変数について説明します。Oracle Databaseをインストールする前に、いくつかの環境変数を定義する必要があります。このセクションでは、次のトピックについて説明します。

- Oracle Databaseの環境変数
- LinuxおよびUNIXの環境変数
- 共通の環境設定
- システム・タイム・ゾーンの設定

環境変数の現在の設定値を表示するには、envコマンドを使用します。たとえば、環境変数ORACLE_SIDの値を表示するには、次のコマンドを実行します。

```
$ env | grep ORACLE_SID
```

すべての環境変数の現在の設定値を表示するには、envコマンドを次のように実行します。

```
$ env | more
```

1.2.1 Oracle Databaseの環境変数

次の表は、Oracle Databaseで使用する一部の環境変数を示します。

表1-1 Oracle Databaseの環境変数

変数	定義
NLS_LANG	機能: クライアント環境の言語、地域および文字セットを指定します。NLS_LANG で指定するクライアント文字セットは、端末または端末エミュレータの文字セットと一致している必要があります。必要に応じて、非対話型バッチ・プログラムを起動する前に NLS_LANG を一時的にリセットして別の文字セットに変更し、そのプログラムで処理されるファイルおよびスクリプトの文字セットに一致させることができます。NLS_LANG で指定された文字セットがデータベースの文字セットと異なる場合、その文字セットは自動的に変換されます。 この変数のパラメータ・リストの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。 構文: language_territory.characterset 例: french_france.we8iso8859p15
ORA_NLS10	機能: 言語、地域、文字セットおよび言語の定義ファイルが保存されているディレクトリを指定します。 構文: directory_path 例: $ORACLE_HOME/nls/data
ORA_TZFILE	機能: タイム・ゾーン・ファイルのフルパスおよびファイル名を指定します。Oracle Database Server では、常に大規模タイム・ゾーン・ファイル ($ORACLE_HOME/oracore/zoneinfo/timezlrg_number.dat)が使用されます。クライアント側で小規模タイム・ゾーン・ファイルを使用する場合、この環境変数を小規模タイム・ゾーン・ファイルのフルパス ($ORACLE_HOME/oracore/zoneinfo/timezone_number.dat)に設定する必要があります。クライアント側で小規模タイム・ゾーン・ファイルを使用する場合、アクセスするデータベースでは、必ず小規模タイム・ゾーン・ファイルで認識されるタイム・ゾーン・リージョンのみにデータが含まれるようにする必要があります。 構文: directory_path 例: $ORACLE_HOME/oracore/zoneinfo/timezlrg_11.dat

変数	定義
ORACLE_BASE	機能: Optimal Flexible Architecture に準拠したインストールの Oracle ディレクトリ構造の基本となるディレクトリを指定します。 構文: directory_path 例: /u01/app/oracle
ORACLE_HOME	機能: Oracle ソフトウェアが格納されているディレクトリを指定します。 構文: directory_path 例: $ORACLE_BASE/product/19.0.0/dbhome_1
ORACLE_PATH	機能: SQL*Plus などの Oracle アプリケーションが使用するファイルの検索パスを指定します。ファイルのフルパス名が指定されていない場合やファイルが現行のディレクトリにない場合、Oracle アプリケーションでは、ORACLE_PATH を使用してそのファイルの場所を特定します。 構文: ディレクトリをコロンで区切ったリスト: directory1:directory2:directory3 例: /u01/app/oracle/product/19.0.0/dbhome_1/bin:
ORACLE_SID	機能: Oracle システムの識別子を指定します。 構文: 英字で始まり、数字と英字で構成される文字列を指定します。システム識別子は、8 文字以内で指定することをお薦めします。この環境変数の詳細は、『Oracle Database インストレーション・ガイド』を参照してください。 例: SAL1
ORACLE_TRACE	機能: インストール時のシェル・スクリプトのトレースを有効にします。この環境変数を T に設定した場合は、ほとんどの Oracle シェル・スクリプトで set -x コマンドが使用されます。これによって、シェル・スクリプトの実行時にコマンドとそれらの引数が印刷されます。他の値を設定した場合、または値を設定しない場合、そのスクリプトでは、set -x コマンドが使用されません。 構文: T または T 以外 例: T

変数	定義
ORAENV_ASK	機能: oraenv または coraenv スクリプトで、環境変数 ORACLE_SID の値を入力するためのプロンプトを表示するかどうかを制御します。NO に設定した場合、環境変数 ORACLE_SID の値を入力するためのプロンプトは表示されません。他の値に設定した場合、または値を設定しない場合、スクリプトによって環境変数 ORACLE_SID の値を入力するためのプロンプトが表示されます。 構文: NO または NO 以外 例: NO
SQLPATH	機能: SQL*Plus での login.sql ファイルの検索先ディレクトリまたはディレクトリのリストを指定します。 構文: ディレクトリをコロンで区切ったリスト: directory1:directory2:directory3 例: /home:/home/oracle:/u01/oracle
TNS_ADMIN	機能: Oracle Net Services 構成ファイルが格納されているディレクトリを指定します。 構文: directory_path 例: $ORACLE_HOME/network/admin
TWO_TASK	機能: 接続文字列に使用するデフォルトの接続識別子を指定します。この環境変数が設定されている場合は、接続文字列に接続識別子を指定しないでください。たとえば、環境変数 TWO_TASK が sales に設定されている場合、次のコマンドを使用してデータベースに接続できます。 SQL> CONNECT username Enter password: password 構文: 任意の接続識別子 値の範囲: ネーミング・メソッドを使用して解決できる有効な接続識別子 (tnsnames.ora ファイルやディレクトリ・サーバーなど) 例: PRODDB_TCP
NLS_OS_CHARSET	機能: ファイル名およびユーザー名がオペレーティング・システムによりエンコードされる UNIX ロケールの文字セットに対応する Oracle 文字セット名を指定します。UNIX ロケール文字セットと Oracle クライアントの文字セットが異なる場合、環境変数 NLS_OS_CHARSET を設定する必要があります。たとえば、NLS_LANG が SQL スクリプトのエンコードで使用する特定の文字セットに設定されて、そのスクリプトが SQL*Plus

変数	定義
	セッションで実行される場合、これらの 2 つの文字セットは異なる場合があります。通常、Oracle クライアントの文字セットおよびオペレーティング・システムの文字セットは同一で、NLS_OS_CHARSET を設定しなくでください。 構文: characterset 例: WE8ISO8859P1

ノート:

- 競合を防ぐため、Oracle Database サーバーのプロセスと同じ名前の環境変数を定義しないでください。たとえば、ARCH、PMON および DBWR などです。

- Oracle インストール所有者(oracle ユーザー)の環境変数が未設定の場合、または設定したパスが正しくない場合、Oracle Database の起動時に次のような未定義のエラーが発生することがあります。データベースを起動する前に、すべての環境変数が正しく設定されていることを確認します。詳細は、My Oracle Support ノート 373303.1 を参照してください。

  https://support.oracle.com/epmos/faces/DocumentDisplay?id=373303.1

注意:

My Oracle Support ノート 373303.1 で提案されている変更を実装する前に、Oracle Support に連絡する必要があります。

1.2.2 LinuxおよびUNIXの環境変数

Oracle Databaseでは、オペレーティング・システムの環境変数を使用して、システム・リソースとソフトウェアの場所を定義します。

表1-2 Oracle Databaseで使用する環境変数

変数	定義
ADA_PATH (IBM AIX on POWER Systems (64-Bit)のみ)	機能: Ada コンパイラが格納されているディレクトリを指定します 構文: directory_path 例: /usr/lpp/powerada
CLASSPATH	機能: Java アプリケーションで使用します。この環境変数に必要な設定は、Java アプリケーションによって異なります。詳細は、Java アプリケーション製品のドキュメントを参照してくださ

変数	定義

	い。
	
	構文: ディレクトリまたはファイルをコロンで区切ったリスト:
	`directory1:directory2:file1:file1`
	
	例: デフォルトの設定はありません。CLASSPATH には、次のディレクトリが含まれている必要があります。
	
	`$ORACLE_HOME/jdk/jre/lib:$ORACLE_HOME/jlib`

DISPLAY	機能 :X ベースのツールで使用します。入出力に使用するディスプレイ・デバイスを指定します。詳細は、使用している X Window System のドキュメントを参照してください。 構文: `hostname:server[.screen]` hostname はシステム名(IP アドレスまたは別名)、server はサーバーの順次コード番号、screen は画面の順次コード番号です。使用するモニターが 1 つの場合は、サーバーと画面のどちらにも値 0 を使用します(0.0)。 ノート: 使用するモニターが 1 つの場合、screen はオプションです。 例:`192.0.2.1:0.0` `password:0`
HOME	機能: ユーザーのホーム・ディレクトリを指定します。 構文: `directory_path` 例: `/home/example_home`
LANG または LC_ALL	機能: メッセージなどの出力でオペレーティング・システムが使用する言語および文字セットを指定します。また、Oracle Universal Installer、Oracle Database Configuration Assistant などの Java でプログラムされている Oracle ツールでは、ユーザー・インタフェースの言語を判別するためにこの変数を使用することがあります。詳細は、オペレーティング・システムのドキュメントを参照してください。
LD_OPTIONS	機能: デフォルトのリンカー・オプションを指定します。この環境変数の詳細は、ld の man ページを参照してください。
LPDEST (Oracle Solaris のみ)	機能: デフォルトのプリンタの名前を指定します。 構文: `string`

変数	定義
	例: docprinter
LD_LIBRARY_PATH	機能: UNIX および Linux 上のライブラリの検索に使用するパスを指定する環境変数。環境変数の名前は、LIBPATH(IBM AIX on POWER Systems (64-Bit)の場合)、SHLIB_PATH(HP-UX の場合)など、オペレーティング・システムにより異なる場合があります。 構文: ディレクトリをコロンで区切ったリスト: directory1:directory2:directory3 例: /usr/dt/lib:$ORACLE_HOME/lib
PATH	機能: シェルで、実行可能プログラムの場所を特定するために使用されます。$ORACLE_HOME/bin ディレクトリが含まれている必要があります。 構文: ディレクトリをコロンで区切ったリスト: directory1:directory2:directory3 例: /bin:/usr/bin:/usr/local/bin:/usr/bin/X11:$ORACLE_HOME/bin:$HOME/bin:
PRINTER	機能: デフォルトのプリンタの名前を指定します。 構文: string 例: docprinter
TEMP、TMP および TMPDIR	機能: 一時ファイル用のデフォルト・ディレクトリを指定します。設定されている場合、一時ファイルを作成するツールが次のいずれかのディレクトリに作成します。 構文: directory_path 例: /u02/oracle/tmp
USER (HP-UX Itanium システムで接続に SSH を使用する場合)	機能: ログインするユーザーの名前を指定します。 構文: string 例: oracle

1.2.3 共通の環境設定

この項では、デフォルト・シェルに応じてoraenvまたはcoraenvスクリプトを使用し、共通のオペレーティング・システム環境を設定する方法について説明します。

- Bourne、BashまたはKornシェルの場合は、oraenvコマンドを使用します。

- Cシェルの場合は、coraenvコマンドを使用します。

oraenvおよびcoraenvスクリプト・ファイル

oraenvおよびcoraenvスクリプトはインストール中に作成されます。これらのスクリプトは、oratabファイルの内容に基づいて環境変数を設定し、次の機能を提供します。

- データベースの変更をすべてのユーザー・アカウントに反映して更新するための主な方法
- oratabファイルに指定されているデータベース間で切替えを行うためのメカニズム

開発システムからデータベースに対して頻繁に追加や削除を行ったり、同一システム上にインストールされた複数の異なるOracle Database間でユーザーが切替えを行う場合があります。oraenvまたはcoraenvスクリプトを使用すると、ユーザー・アカウントが更新されていることを確認し、データベース間で切替えを行うことができます。

ノート:

oraenv または coraenv スクリプトは、Oracle ソフトウェア所有者(通常は oracle)ユーザーのシェル起動スクリプトからはコールしないでください。これらのスクリプトでは値の入力を促すプロンプトが表示されるため、システムの起動時に dbstart スクリプトが自動的にデータベースを起動できなくなります。

oraenvまたはcoraenvスクリプトは通常、ユーザーのシェル起動ファイル(.profileまたは.loginなど)からコールされます。これは、環境変数ORACLE_SIDおよびORACLE_HOMEを設定し、$ORACLE_HOME/binディレクトリを環境変数PATHの設定に含めます。データベース間で切替えを行う場合に、oraenvまたはcoraenvスクリプトを実行して、これらの環境変数を設定できます。

ノート:

これらのスクリプトのいずれかを実行するには、適切なコマンドを使用します。

- coraenv スクリプトの場合:

```
% source /usr/local/bin/coraenv
```

- oraenv スクリプトの場合:

```
$ . /usr/local/bin/oraenv
```

ローカルbinディレクトリ

oraenv、coraenvおよびdbhomeスクリプトを含むディレクトリは、ローカルbinディレクトリと呼ばれます。すべてのデータベース・ユーザーは、このディレクトリへの読取りアクセス権が必要です。ローカルbinディレクトリのパスをユーザーの環境変数PATHの設定に追加してください。インストール後にroot.shスクリプトを実行すると、ローカルbinディレクトリのパスを要求するプロンプトが表示されます。指定したディレクトリに、oraenv、coraenvおよびdbhomeスクリプトが自動的にコピーされます。デフォルトのローカルbinディレクトリは、/usr/local/binです。root.shスクリプトを実行しない場合は、手動でoraenvまたはcoraenvスクリプトとdbhomeスクリプトを、$ORACLE_HOME/binディレクトリからローカルbinディレクトリにコピーできます。

1.2.4 システム・タイム・ゾーンの設定

環境変数TZは、タイム・ゾーンを設定します。これによって、時間を夏時間に変更したり、別のタイム・ゾーンにすることができます。

関連項目:

- "表1-1"の"ORA_TZFILE"
- データベース・タイムゾーンの設定の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』および『Oracle Database管理者ガイド』を参照してください。

1.3 初期化パラメータ

これらの各項では、Oracle Database初期化パラメータについて説明します。

トピック:

- ASM_DISKSTRING初期化パラメータ
- DISK_ASYNCH_IO初期化パラメータ(HP-UX)
- PROCESSOR_GROUP_NAME初期化パラメータ
- 診断データの管理

1.3.1 ASM_DISKSTRING初期化パラメータ,

ノート:

ASM_DISKSTRING 初期化パラメータをサポートしているのは、自動ストレージ管理インスタンスのみです。

ASM_DISKSTRING初期化パラメータに値を割り当てるための構文は、次のとおりです。

```
ASM_DISKSTRING = 'path1'[,'path2', . . .]
```

この構文のpathnは、RAWデバイスへのパスです。パスを指定する際は、ワイルドカード文字を使用できます。

表1-3に、ASM_DISKSTRING初期化パラメータに対するプラットフォーム固有のデフォルト値を示します。

表1-3 ASM_DISKSTRING初期化パラメータのデフォルト値

プラットフォーム	デフォルト検索文字列
Oracle Solaris	/dev/rdsk/*
Linux	/dev/sd*
IBM AIX on POWER Systems	/dev/rhdisk*

プラットフォーム	デフォルト検索文字列
(64 ビット)	
HP-UX	/dev/rdisk*

1.3.2 DISK_ASYNCH_IO初期化パラメータ(HP-UX)

DISK_ASYNCH_IO初期化パラメータは、データベース・ファイルがRAWディスクとファイル・システムのどちらに置かれるかを決定します。非同期I/Oは、データベース・ファイルの記憶域オプションとしてRAWパーティションを使用する自動ストレージ管理ディスク・グループでのみ使用できます。DISK_ASYNCH_IOパラメータには、ファイルが存在する場所に応じてTRUEまたはFALSEを設定できます。デフォルトでは、この値はTRUEに設定されます。

ノート:

DISK_ASYNCH_IO パラメータは、データベース・ファイルがファイル・システム上に置かれる場合、FALSE に設定する必要があります。このパラメータを TRUE に設定する必要があるのは、データベース・ファイルが RAW パーティション上に置かれる場合のみです。

1.3.3 PROCESSOR_GROUP_NAME初期化パラメータ

PROCESSOR_GROUP_NAMEは、インスタンスが稼働しているプロセッサ・グループの名前を指定します。このパラメータは、指定されたオペレーティング・システム・プロセッサ・グループのプロセッサでのみ稼働するよう、Oracle Databasesに対して指示します。NUMAシステムの場合、システム・グローバル領域(SGA)およびプログラム・グローバル領域(PGA)はすべて、このプロセッサ・グループのCPUに関連付けられたNUMAノードから割り当てられます。

PROCESSOR_GROUP_NAMEパラメータはLinux x86-64およびOracle Solaris 11 SRU 4以上でのみサポートされています。

Linux x86-64では、制御グループ(cgroups)と呼ばれるLinuxの機能によってCPUの名前付きサブセットが作成されます。cgroupsはLinuxカーネル・バージョン2.6.24で導入されます。これは、グループの名前と1つのCPUセットを指定して作成します。プロセスがcgroupにマップされると、そのcgroupに関連付けられたCPUのみが使用されます。

Oracle Solaris 11 SRU 4では、リソース・プールと呼ばれる機能を使用してCPUの名前付きサブセットが作成されます。各リソース・プールは名前と1つのCPUセットで構成されます。プロセスがリソース・プールにマップされると、そのリソース・プールに関連付けられたCPUが使用されます。

ノート:

PROCESSOR_GROUP_NAME パラメータは、専用の接続ブローカを使用するデータベースに対してのみ設定することをお薦めします。専用接続ブローカの構成には、USE_DEDICATED_BROKER 初期化パラメータを使用します。

関連項目

- Oracle Databaseリファレンス

- Oracle Databaseリファレンス

1.3.4 診断データの管理

診断データにはトレース・ファイル、ダンプおよびコア・ファイルがあり、問題を短時間で効率的に解決できます。

関連項目

- Oracle Database管理者ガイド

2 Oracleソフトウェアの停止と起動

この章では、Oracle Databaseプロセスを識別する方法と、プロセスを停止および再起動する基本的な方法について説明します。また、Oracle Databaseの起動と停止を自動化する方法についても説明します。内容は次のとおりです。

- Oracleプロセスの停止と起動
- 停止と起動の自動化

ノート:

Oracle Restart を使用する場合、コマンドライン・インタフェースであるサービス制御ユーティリティ(SRVCTL)を使用して、Oracle プロセス(データベース・インスタンス、リスナー、Oracle ASM インスタンス)を管理できます。SRVCTL を使用すると、Oracle Restart 構成の管理、Oracle Restart で管理されるプロセスのステータスの確認、および Oracle Database などのプロセスの開始や停止が可能です。SRVCTL は Oracle Clusterware と、Oracle Restart を使用した 1 インスタンスの Oracle Database をサポートするように拡張されています。

関連項目:

SRVCTLコマンドの詳細は、『Oracle Database管理者ガイド』と『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

2.1 Oracleプロセスの停止と起動

この項では、Oracleプロセスを停止および起動する方法について説明します。次の項目が含まれます。

- Oracle DatabaseインスタンスおよびOracle Automatic Storage Managementインスタンスの停止と起動
- Oracle Restartの停止と起動

2.1.1 Oracle DatabaseインスタンスおよびOracle Automatic Storage Managementインスタンスの停止と起動

この項では、Oracle DatabaseインスタンスおよびOracle Automatic Storage Managementインスタンスを停止および起動する方法について説明します。内容は次のとおりです。

- Oracle DatabaseインスタンスまたはOracle Automatic Storage Managementインスタンスの停止
- Oracle DatabaseインスタンスまたはOracle Automatic Storage Managementインスタンスの再起動

2.1.1.1 Oracle DatabaseインスタンスまたはOracle Automatic Storage Managementインスタンスの停止

注意:

Oracle Automatic Storage Management インスタンスを使用して記憶域を管理している Oracle

Database インスタンスをすべて停止するまで、Oracle Automatic Storage Management インスタンスは停止しないでください。

Oracle DatabaseインスタンスまたはOracle Automatic Storage Managementインスタンスを停止するには:

1. 次のコマンドを実行して、停止する必要があるインスタンスのSIDおよびOracleホーム・ディレクトリを識別します。

   Oracle Solarisの場合:

   ```
   $ cat /var/opt/oracle/oratab
   ```

   その他のオペレーティング・システムの場合

   ```
   $ cat /etc/oratab
   ```

   oratabファイルには、次のような行が含まれています。これによって、システム上の各データベース・インスタンスまたはOracle Automatic Storage ManagementインスタンスのSIDおよび対応するOracleホーム・ディレクトリを識別します。

   ```
   $ORACLE_SID:$ORACLE_HOME:<N/Y>
   ```

   ノート:

   Oracle Automatic Storage Management インスタンスの SID には、1 文字目にプラス記号(+)を使用することをお薦めします。

2. デフォルト・シェルに応じてoraenvまたはcoraenvスクリプトを実行し、停止する必要があるインスタンスの環境変数を設定します。
   - Bourne、BashまたはKornシェル:

     ```
     $ . /usr/local/bin/oraenv
     ```

   - Cシェルの場合:

     ```
     % source /usr/local/bin/coraenv
     ```

   プロンプトが表示されたら、インスタンスのSIDを指定します。

3. 次のコマンドを実行し、インスタンスを停止します。

   ```
   $ sqlplus
   SQL> CONNECT SYS AS SYSDBA
   Enter password: sys_password
   SQL> SHUTDOWN NORMAL
   ```

   インスタンスの停止後、SQL*Plusを終了できます。

2.1.1.2 Oracle DatabaseインスタンスまたはOracle Automatic Storage Managementインスタンスの再起動

注意:

データベース・インスタンスで記憶域管理に Oracle Automatic Storage Management を使用している場

合は、データベース・インスタンスを起動する前に、Oracle Automatic Storage Management インスタンスを起動する必要があります。

Oracle DatabaseインスタンスまたはOracle Automatic Storage Managementインスタンスを再起動するには:

1. 必要に応じて、ステップ1と2を繰り返し、環境変数ORACLE_SIDおよびORACLE_HOMEを設定して、起動するインスタンスのSIDおよびOracleホーム・ディレクトリを識別します。
2. 次のコマンドを実行し、インスタンスを起動します。

```
$ sqlplus
SQL> CONNECT SYS AS SYSDBA
Enter password: sys_password
SQL> STARTUP
```

インスタンスの起動後、SQL*Plusを終了できます。

2.1.2 Oracle Restartの停止と起動

Oracle Restartを停止または起動するには、次のコマンドを実行します。

- 起動: このオプションを使用して、Oracle Restartを起動します。

  構文とオプション:

```
crsctl start has
```

- 停止: このオプションを使用して、Oracle Restartを停止します。

  構文とオプション:

```
crsctl stop has
```

関連項目:

srvctlコマンドの詳細は、『Oracle Database管理者ガイド』を参照してください。

2.2 データベースの起動と停止の自動化について

Oracle Databaseは、システムの起動時に自動的に起動し、停止時に自動的に停止するようにシステムを構成することをお薦めします。データベースの起動と停止を自動化することによって、データベースの不正な停止を防ぐことができます。

データベースの起動と停止を自動化するには、$ORACLE_HOME/binディレクトリにあるdbstartおよびdbshutスクリプトを使用します。これらのスクリプトは、oratabファイル内の同じエントリを参照します。したがって、同じデータベース・セットに適用されます。たとえば、dbstartスクリプトによって、sid1、sid2およびsid3を自動的に起動し、dbshutスクリプトによって、sid1のみを停止することはできません。ただし、dbstartスクリプトがまったく使用されていない場合は、dbshutスクリプトを使用してデータベース・セットの停止を指定することはできます。これを実行するにはシステムの停止ファイルにdbshutエントリを含めますが、システムの起動ファイルにdbstartエントリは含めないでください。

関連項目:

システムの起動と停止の手順については、オペレーション・システムのドキュメントにあるinitコマンドを参照してください。

2.2.1 データベースの起動と停止の自動化

dbstartおよびdbshutスクリプトを使用してデータベースの起動と停止を自動化するには:

1. rootユーザーでログインします。
2. プラットフォームのoratabファイルを編集します。

   ファイルを開くには、次のいずれかのコマンドを使用します。

   - Oracle Solarisの場合:

     ```
     # vi /var/opt/oracle/oratab
     ```

   - IBM AIX on POWER Systems (64-Bit)およびLinuxの場合:

     ```
     # vi /etc/oratab
     ```

   oratabファイル内のデータベース・エントリは、次の形式で表示されます。

   ```
   $ORACLE_SID:$ORACLE_HOME:<N|Y>
   ```

   この例の値YおよびNは、スクリプトでデータベースの起動または停止を実行するかどうかを指定します。最初に、停止と起動を自動化するデータベースごとに、データベースのインスタンス識別子(SID)を判別します。これは、最初のフィールドのSIDで識別されます。次に、最後のフィールドをそれぞれYに変更します。

   dbstartを設定すると、自動ストレージ管理インストールを使用している単一インスタンスのデータベースの起動を自動化できます。自動ストレージ管理はOracle Clusterwareによって自動的に起動されます。これは、自動ストレージ管理クラスタのデフォルトの動作です。これを実行する場合は、データベースおよび自動ストレージ管理インストールのoratabエントリを変更して、3番目のフィールドに値WおよびNをそれぞれ設定する必要があります。これらの値は、自動ストレージ管理インスタンスの起動後にのみ、dbstartがデータベースを自動的に起動するように指定します。

   > ノート:
   >
   > 新規データベース・インスタンスをシステムに追加する場合、これらのインスタンスを自動的に起動するには、oratab ファイルでインスタンスのエントリを編集する必要があります。

3. オペレーティング・システムに応じ、ディレクトリを次のいずれかに変更します。

プラットフォーム	初期化ファイルのディレクトリ
Linux および Oracle Solaris	/etc/init.d
IBM AIX on POWER Systems (64 ビット)	/etc

4. dboraというファイルを作成し、次の行をこのファイルにコピーします。

   > ノート:
   >
   > 環境変数 ORA_HOME の値を変更し、インストールの Oracle ホーム・ディレクトリに指定します。また、環

境変数 ORA_OWNER の値を、Oracle ホーム・ディレクトリにインストールされているデータベースの所有者のユーザー名(通常は oracle)に変更します。

```
#! /bin/sh
```

description: Oracle auto start-stop script. # # Set ORA_HOME to be equivalent to the $ORACLE_HOME # from which you wish to execute dbstart and dbshut; # # Set ORA_OWNER to the user id of the owner of the # Oracle database in ORACLE_HOME. ORA_HOME=<Type your ORACLE_HOME in full path here> ORA_OWNER=<Type your Oracle account name here> case "$1" in 'start') # Start the Oracle databases: # The following command assumes that the oracle login # will not prompt the user for any values # Remove "&" if you don't want startup as a background process. su - $ORA_OWNER -c "$ORA_HOME/bin/dbstart $ORA_HOME" & touch /var/lock/subsys/dbora ;; 'stop') # Stop the Oracle databases: # The following command assumes that the oracle login # will not prompt the user for any values su - $ORA_OWNER -c "$ORA_HOME/bin/dbshut $ORA_HOME" & rm -f /var/lock/subsys/dbora ;; esac

ノート:

このスクリプトで停止できるのは、パスワードが設定されていない Oracle Net Listener のみです。また、リスナー名がデフォルトの LISTENER ではない場合、stop および start コマンドでリスナー名を指定する必要があります。

```
$ORACLE_HOME/bin/lsnrctl {start|stop} listener_name
```

5. doraファイルのグループをOSDBAグループ(通常はdba)に変更し、その権限を750に設定します。

```
# chgrp dba dbora
# chmod 750 dbora
```

次のように、doraスクリプトへのシンボリック・リンクを、適切な起動レベルのスクリプト・ディレクトリに作成します。

プラットフォーム	シンボリック・リンク・コマンド
Oracle Solaris	# ln -s /etc/init.d/dbora /etc/rc0.d/K01dbora # ln -s /etc/init.d/dbora /etc/rc3.d/S99dbora
Linux	# ln -s /etc/init.d/dbora /etc/rc.d/rc0.d/K01dbora # ln -s /etc/init.d/dbora /etc/rc.d/rc3.d/S99dbora # ln -s /etc/init.d/dbora /etc/rc.d/rc5.d/S99dbora
IBM AIX on POWER Systems (64 ビット)	# ln -s /etc/dbora /etc/rc.d/rc2.d/S99dbora # ln -s /etc/dbora /etc/rc.d/rc0.d/K01dbora

3 Oracle Databaseの構成

この章では、Oracle製品に対するOracle Databaseの構成方法について説明します。内容は次のとおりです。

- コンフィギュレーション・アシスタントの使用
- 実行可能ファイルの再リンク

3.1 コンフィギュレーション・アシスタントの使用

Oracle Databaseソフトウェアには、様々なデータベース管理操作の実行に使用できるコンフィギュレーション・アシスタントが付属しています。

この項では、次の項目について説明します。

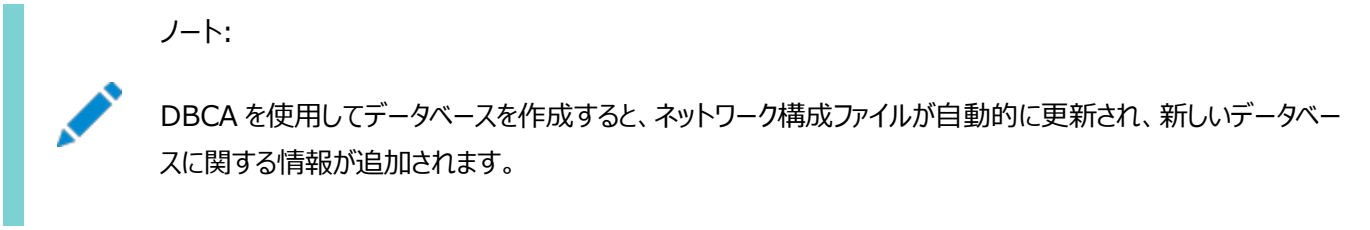
- Oracle Net Configuration Assistantの使用
- Oracle Database Upgrade Assistantの使用
- Oracle Database Configuration Assistantの使用

3.1.1 Oracle Net Configuration Assistantの使用

Oracle Net Configuration Assistantでは、リスナー名、プロトコル・アドレス、ネーミング・メソッド、`tnsnames.ora`ファイル内のネット・サービス名、ディレクトリ・サーバーの使用などの基本ネットワーク・コンポーネントをインストール時に構成できます。

インストールの完了後、Oracle Net Configuration Assistantを使用して詳細な構成を作成できます。次のコマンドを入力します。

```
$ $ORACLE_HOME/bin/netca
```



ノート:

DBCA を使用してデータベースを作成すると、ネットワーク構成ファイルが自動的に更新され、新しいデータベースに関する情報が追加されます。

3.1.2 Oracle Database Upgrade Assistantの使用

Oracle Database Upgrade Assistant (DBUA)を使用すると、対話形式のガイドに従ってデータベースをアップグレードし、新規リリースに応じてデータベースを構成できます。DBUAでは、通常は手動で実行されるタスクをすべて実行することによって、アップグレードを自動化します。DBUAでは、表領域やオンラインREDOログなどの構成オプションに対して推奨設定を作成します。

DBUAを起動するには、次のコマンドを実行します。

```
$ $ORACLE_HOME/bin/dbua
```

DBUAで使用できるコマンドライン・オプションに関する情報を取得するには、次のように、`-help`または`-h`コマンドライン引数を使用します。たとえば:

```
$ $ORACLE_HOME/bin/dbua -help
```

関連項目

- Oracle Databaseアップグレード・ガイド

3.1.3 Oracle Database Configuration Assistantの使用

Oracle Database Configuration Assistant (DBCA)は、デフォルトまたはカスタマイズされたデータベースの作成および構成を支援するグラフィカル・ユーザー・インタフェースです。また、既存のデータベースを構成してOracle Database機能に追加し、Oracle Automatic Storage Managementディスク・グループを作成することもできます。後でデータベース作成時に検査、修正および実行できるシェル・スクリプトとSQLスクリプトのセットもDBCAで生成します。

DBCAを起動するには、次のコマンドを実行します。

```
$ $ORACLE_HOME/bin/dbca
```

DBCAで使用できるコマンドライン・オプションに関する情報を取得するには、次のように、`-help`または`-h`コマンドライン引数を使用します。たとえば:

```
$ $ORACLE_HOME/bin/dbca -help
```

3.2 実行可能ファイルの再リンク

`$ORACLE_HOME/bin`ディレクトリ内の`relink`シェル・スクリプトを使用して、製品の実行可能ファイルを手動で再リンクできます。製品の実行可能ファイルの再リンクは、オペレーティング・システムのパッチを適用するたびに、またはオペレーティング・システムのアップグレード後に必要となります。

> ノート:
>
> 実行可能ファイルの再リンク前に、Oracle ホーム・ディレクトリで実行されている、再リンクする実行可能ファイルをすべて停止する必要があります。また、Oracle 共有ライブラリにリンクされているアプリケーションも停止してください。`relink` スクリプトは引数として `all` および `as_installed` をとります。パラメータを指定しない場合は、`all` 引数とみなされます。

`relink`スクリプトを使用すると、すべてのOracle製品の実行可能ファイルを再リンクできますが、その方法はOracleホーム・ディレクトリにインストールされている製品によって異なります。

関連項目:

自動ストレージ・マネージャでrelinkスクリプトを使用する方法の詳細は、『Oracle Databaseインストレーション・ガイド for Linux』を参照してください。

製品の実行可能ファイルを再リンクするには、次のコマンドを実行します。

```
$ relink
```

4 SQL*Plusの管理

この章では、SQL*Plusの管理方法について説明します。内容は次のとおりです。

- コマンドラインSQL*Plusの管理
- コマンドラインSQL*Plusの使用
- SQL*Plusの制限事項

関連項目

- SQL*Plusユーザーズ・ガイドおよびリファレンス

4.1 コマンドラインSQL*Plusの管理

この項では、コマンドラインSQL*Plusの管理方法について説明します。例では、SQL*Plusにより、疑問符(?)は環境変数ORACLE_HOMEの値に置き換えられています。

- 設定ファイルの使用
- Oracle Databaseサンプル・スキーマの使用
- SQL*Plusのコマンドライン・ヘルプのインストールと削除

4.1.1 設定ファイルの使用

SQL*Plusを起動すると、最初にサイト・プロファイル設定ファイルglogin.sqlが実行され、次にユーザー・プロファイル設定ファイルlogin.sqlが実行されます。

サイト・プロファイル・ファイルの使用

グローバルなサイト・プロファイル・ファイルは、$ORACLE_HOME/sqlplus/admin/glogin.sqlです。この場所にサイト・プロファイル・ファイルがすでに存在する場合は、SQL*Plusのインストール時にそのファイルが上書きされます。SQL*Plusを削除すると、そのサイト・プロファイル・ファイルも削除されます。

ユーザー・プロファイル・ファイルの使用

ユーザー・プロファイル・ファイルは、login.sqlです。SQL*Plusは、最初に現行のディレクトリを検索し、次に環境変数ORACLEPATHで指定したディレクトリを検索してこのファイルを検出します。この環境変数の値は、ディレクトリをコロンで区切ったリストです。SQL*Plusでは、これらのディレクトリを環境変数ORACLEPATHにリストされている順序で検索し、login.sqlファイルを検出します。

login.sqlファイルに設定されているオプションは、glogin.sqlファイルに設定されているオプションよりも優先されます。

関連項目:

プロファイル・ファイルの詳細は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください。

4.1.2 Oracle Databaseサンプル・スキーマの使用

Oracle Databaseをインストールするか、またはOracle Database Configuration Assistantを使用してデータベースを作成する場合は、Oracle Databaseサンプル・スキーマをインストールできます。

関連項目

- Oracle Databaseサンプル・スキーマ

4.1.3 SQL*Plusのコマンドライン・ヘルプのインストールと削除

この項では、SQL*Plusのコマンドライン・ヘルプのインストール方法と削除方法について説明します。

- SQL*Plusのコマンドライン・ヘルプのインストール
- SQL*Plusのコマンドライン・ヘルプの削除

関連項目

- SQL*Plusユーザーズ・ガイドおよびリファレンス

4.1.3.1 SQL*Plusのコマンドライン・ヘルプのインストール

SQL*Plusのコマンドライン・ヘルプは、次の3つの方法でインストールできます。

- 事前構成済データベースのインストールを完了します。

  インストールの一部として事前構成済データベースをインストールすると、SQL*Plusのコマンドライン・ヘルプがSYSTEMスキーマに自動的にインストールされます。

- `$ORACLE_HOME/bin/helpins`シェル・スクリプトを使用して、SYSTEMスキーマにコマンドライン・ヘルプを手動でインストールします。

  `helpins`スクリプトは、SYSTEMパスワードを要求するプロンプトを表示します。プロンプトを表示せずにこのスクリプトを実行する場合は、環境変数`SYSTEM_PASS`にSYSTEMユーザー名とパスワードを設定します。たとえば:

  - Bourne、BashまたはKornシェル:

    ```
    $ SYSTEM_PASS=SYSTEM/system_password; export SYSTEM_PASS
    ```

  - Cシェルの場合:

    ```
    % setenv SYSTEM_PASS SYSTEM/system_password
    ```

- `$ORACLE_HOME/sqlplus/admin/help/helpbld.sql`スクリプトを使用して、SYSTEMスキーマにコマンドライン・ヘルプを手動でインストールします。

  たとえば、次のコマンドを実行します。`system_password`は、SYSTEMユーザーのパスワードです。

  ```
  $ sqlplus
  SQL> CONNECT SYSTEM
  Enter password: system_password
  SQL> @?/sqlplus/admin/help/helpbld.sql ?/sqlplus/admin/help helpus.sql
  ```

> ノート:
>
> `helpins` シェル・スクリプトおよび `helpbld.sql` スクリプトは、新しい表を作成する前に、既存のコマンドライン・ヘルプの表を削除します。

4.1.3.2 SQL*Plusのコマンドライン・ヘルプの削除

SYSTEMスキーマからSQL*Plusのコマンドライン・ヘルプの表を手動で削除するには、$ORACLE_HOME/sqlplus/admin/help/helpdrop.sqlスクリプトを実行します。このスクリプトには、次のコマンドを実行します。system_passwordは、SYSTEMユーザーのパスワードです。

```
$ sqlplus
SQL> CONNECT SYSTEM
Enter password: system_password
SQL> @?/sqlplus/admin/help/helpdrop.sql
```

4.2 コマンドラインSQL*Plusの使用

この項では、コマンドラインSQL*Plusの使用方法について説明します。次の項目が含まれます。

- SQL*Plusからのシステム・エディタの使用
- SQL*Plusからのオペレーティング・システム・コマンドの実行
- SQL*Plusへの割込み
- SPOOLコマンドの使用

4.2.1 SQL*Plusからのシステム・エディタの使用

SQL*PlusプロンプトでEDまたはEDITコマンドを実行すると、ed、emacs、ned、viなどのオペレーティング・システム・エディタが起動します。ただし、環境変数PATHには、エディタの実行可能ファイルが格納されているディレクトリを指定する必要があります。

HISTORYコマンドを使用すると、以前使用したSQL*Plus、SQLまたはPL/SQLコマンドを現在のセッションの履歴リストから、実行、編集または削除できます。現在のSQL*Plusセッションで発行されたコマンドの履歴を有効または無効にできます。

エディタを起動すると、現行のSQLバッファがエディタに格納されます。エディタを終了すると、変更されたSQLバッファがSQL*Plusに戻されます。

SQL*Plusの_EDITOR変数を定義することにより、起動するエディタを指定できます。この変数は、glogin.sqlサイト・プロファイルまたはlogin.sqlユーザー・プロファイルに定義できます。また、SQL*Plusセッション時に定義することもできます。たとえば、デフォルト・エディタをviに設定するには、次のコマンドを実行します。

```
SQL> DEFINE _EDITOR=vi
```

_EDITOR変数を設定しない場合は、環境変数EDITORまたはVISUALのいずれかの値が使用されます。両方の環境変数が設定されている場合は、環境変数EDITORの値が使用されます。_EDITOR、EDITORおよびVISUALのいずれも指定されていない場合、デフォルト・エディタはviになります。定義されたエディタは、HISTORYコマンドの編集オプションで使用されます。次のSQL文について考えてみます。

```
SQL> hist
1 select * from dual;
2 desc dual
```

前述のSQL文の最初のエントリをviエディタで開くには、次のコマンドを使用します。

```
SQL> hist 1 edit
```

エディタを起動すると、SQL*Plusは一時ファイルafiedt.bufを使用してエディタにテキストを渡します。SET EDITFILEコ

マンドを使用すると、別のファイル名を指定できます。たとえば:

```
SQL> SET EDITFILE /tmp/myfile.sql
```

SQL*Plusは、一時ファイルを削除しません。

4.2.2 SQL*Plusからのオペレーティング・システム・コマンドの実行

SQL*Plusプロンプトの後の最初の文字としてHOSTコマンドまたは感嘆符(!)を使用すると、後続の文字がサブシェルに渡されます。オペレーティング・システム・コマンドを実行するときに使用するシェルは、環境変数SHELLによって設定されます。デフォルト・シェルはBourneシェルです。シェルが実行できない場合は、SQL*Plusによりエラー・メッセージが表示されます。

SQL*Plusに戻るには、exitコマンドを実行するか、[Ctrl]+[D]を押します。

たとえば、1つのコマンドを実行するには、次のコマンド構文を使用します。

```
SQL> ! command
```

この例のcommandは、実行するオペレーティング・システム・コマンドです。

SQL*Plusから複数のオペレーティング・システム・コマンドを実行するには、HOSTまたは!コマンドを実行します。オペレーティング・システム・プロンプトに戻るには、[Enter]を押します。

4.2.3 SQL*Plusへの割込み

SQL*Plusの実行中、[Ctrl]+[C]を押すと、レコードのスクロール表示を停止したり、SQL文を終了できます。

4.2.4 SPOOLコマンドの使用

SPOOLコマンドで生成されるファイルのデフォルトのファイル名拡張子は、.lstです。この拡張子を変更するには、ピリオド(.)を含めたスプール・ファイル名を指定します。例:

```
SQL> SPOOL query.txt
```

4.3 SQL*Plusの制限事項

この項では、次のSQL*Plusの制限事項を説明します。

- ウィンドウのサイズ変更
- リターン・コード
- パスワードの非表示

4.3.1 ウィンドウのサイズ変更

SQL*PlusのシステムLINESIZEおよびPAGESIZEのデフォルト値では、ウィンドウのサイズは自動的に調整されません。ウィンドウのサイズが変更された場合は、LINESIZEおよびPAGESIZEシステム変数を設定する必要があります。

4.3.2 リターン・コード

オペレーティング・システムのリターン・コードは1バイトですが、Oracleエラー・コードを返すには、1バイトでは不十分です。リターン・コードの範囲は、0から255です。

4.3.3 パスワードの非表示

コマンドラインでパスワードを渡したり、SYSTEM_PASS環境変数をSYSTEMユーザーのユーザー名とパスワードに設定すると、この情報がpsコマンドの出力に表示される場合があります。権限のないアクセスを防ぐため、SQL*Plusによってプロンプトが表示された場合のみSYSTEMパスワードを入力してください。

スクリプトを自動的に実行する場合は、パスワードの格納が不要な認証方法の使用を考慮してください。たとえば、Oracle Databaseへの外部認証ログインなどがあります。セキュリティの低い環境では、スクリプト・ファイルにオペレーティング・システム・パイプを使用して、パスワードをSQL*Plusに渡すことを検討する必要があります。たとえば:

```
$ echo system_password | sqlplus SYSTEM @MYSCRIPT
```

あるいは、次のコマンドを実行します。

```
$ sqlplus <<EOF
SYSTEM/system_password
SELECT ...
EXIT
EOF
```

この例のsystem_passwordは、SYSTEMユーザーのパスワードです。

5 Oracle Net Servicesの構成

この章では、Oracle Net Servicesを構成する方法について説明します。内容は次のとおりです。

- Oracle Net Services構成ファイルの保存場所
- アダプタ・ユーティリティの使用
- Oracleプロトコル・サポートの使用
- TCP/IPまたはSecure Sockets Layer付きTCP/IP用のリスナーの設定

関連項目

- Oracle Database Net Services管理者ガイド

5.1 Oracle Net Services構成ファイルの保存場所

Oracle Net Services構成ファイルは、多くの場合、`$ORACLE_HOME/network/admin`ディレクトリにあります。ファイルのタイプに応じて、Oracle Netは異なる検索順序でファイルを検出します。

`sqlnet.ora`および`ldap.ora`の各ファイルの検索順序は、次のとおりです。

1. 環境変数`TNS_ADMIN`で指定したディレクトリ(この環境変数が設定されている場合)
2. `$ORACLE_HOME/network/admin`ディレクトリ

`cman.ora`、`listener.ora`および`tnsnames.ora`の各ファイルの検索順序は、次のとおりです。

1. 環境変数`TNS_ADMIN`で指定したディレクトリ(この環境変数が設定されている場合)
2. 次のいずれかのディレクトリ
   - Oracle Solarisの場合:

     ```
     /var/opt/oracle
     ```

   - その他のプラットフォームの場合:

     ```
     /etc
     ```

3. `$ORACLE_HOME/network/admin`ディレクトリ

一部のシステム・レベルの構成ファイルでは、対応するユーザー・レベルの構成ファイル(ユーザーのホーム・ディレクトリに格納されている)を作成できます。ユーザー・レベルのファイルの設定によって、システム・レベルのファイルの設定が上書きされます。次の表に、システム・レベルの構成ファイルとそれに対応するユーザー・レベルの構成ファイルを示します。

システム・レベルの構成ファイル	ユーザー・レベルの構成ファイル
`sqlnet.ora`	`$HOME/.sqlnet.ora`
`tnsnames.ora`	`$HOME/.tnsnames.ora`

例5-1 サンプル構成ファイル

`$ORACLE_HOME/network/admin/samples`ディレクトリには、`cman.ora`、`listener.ora`、`sqlnet.ora`、`tnsnames.ora`の各構成ファイルのサンプルが含まれています。

ノート:

- cman.ora ファイルがインストールされるのは、クライアントのカスタム・インストール時にカスタム・オプションの一部として Connection Manager を選択した場合のみです。
- 読取り専用 Oracle ホームでは、構成ファイルは$ORACLE_BASE_HOME/network/admin ディレクトリにあります。

5.2 アダプタ・ユーティリティの使用

adaptersユーティリティには、システムでOracle Databaseがサポートしているトランスポート・プロトコル、ネーミング・メソッドおよびOracle Advanced Securityのオプションが表示されます。

関連項目:

adaptersユーティリティの詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。

5.3 Oracleプロトコル・サポートの使用

Oracle Protocol Supportは、Oracle Netのコンポーネントの1つです。次が含まれています。

- IPCプロトコル・サポート
- TCP/IPプロトコル・サポート
- Secure Sockets Layer付きTCP/IPプロトコル・サポート

IPC、TCP/IPおよびSecure Sockets Layer付きTCP/IPの各プロトコル・サポートには、それぞれアドレス指定が必要です。このアドレスは、Oracle Net Services構成ファイルおよびDISPATCHER初期化パラメータで使用されます。次の各項では、各プロトコル・サポートのアドレス指定について説明します。

関連項目

- Oracle Database Net Services管理者ガイド

5.3.1 IPCプロトコル・サポート

IPCプロトコル・サポートは、クライアント・プログラムとOracle Databaseが同じシステムにインストールされている場合にのみ使用できます。このプロトコル・サポートには、リスナーが必要です。これは、すべてのクライアント・ツールおよびoracle実行可能ファイルにインストールおよびリンクされます。

IPCプロトコル・サポートには、次の書式のアドレス指定が必要です。

```
(ADDRESS = (PROTOCOL=IPC)(KEY=key))
```

次の表に、このアドレス指定で使用されるパラメータを示します。

パラメータ	説明
PROTOCOL	使用するプロトコルを表します。この値は IPC です。大/小文字は区別されません。

パラメータ	説明
KEY	同じシステムで IPC KEY として使用されている他の名前とは異なる、一意の名前を表します。

次に、IPCプロトコル・アドレスのサンプルを示します。

```
(ADDRESS= (PROTOCOL=IPC)(KEY=EXTPROC))
```

5.3.2 TCP/IPプロトコル・サポート

TCP/IPは、ネットワークを介したクライアント/サーバー通信に使用される標準的な通信プロトコルです。TCP/IPプロトコル・サポートを使用すると、クライアント・プログラムとOracle Databaseが同じシステムまたは別のシステムのどちらにインストールされていても、それらの間で通信を行うことができます。システムにTCP/IPプロトコルをインストールした場合、TCP/IPプロトコル・サポートがすべてのクライアント・ツールおよびoracle実行可能ファイルにインストールおよびリンクされます。

TCP/IPプロトコル・サポートには、次の書式のアドレス指定が必要です。

```
(ADDRESS = (PROTOCOL=TCP)(HOST=hostname)(PORT=port))
```

次の表に、このアドレス指定で使用されるパラメータを示します。

パラメータ	説明
PROTOCOL	使用するプロトコル・サポートを表します。この値は TCP です。大/小文字は区別されません。
HOST	ホスト名またはホストの IP アドレスを表します。
PORT	TCP/IP ポートを表します。このポートには、番号または/etc/services ファイルでこのポートにマップされた別名を指定します。推奨値は 1521 です。

次に、TCP/IPプロトコル・アドレスのサンプルを示します。

```
(ADDRESS= (PROTOCOL=TCP)(HOST=MADRID)(PORT=1521))
```

5.3.3 Secure Sockets Layer付きTCP/IPプロトコル・サポート

Secure Sockets Layer付きTCP/IPプロトコル・サポートを使用すると、クライアントのOracleアプリケーションは、TCP/IPおよびSecure Sockets Layerを介したリモートのOracle Databaseインスタンスと通信できます。

Secure Sockets Layer付きTCP/IPプロトコル・サポートには、次の書式のアドレス指定が必要です。

```
(ADDRESS = (PROTOCOL=TCPS)(HOST=hostname)(PORT=port))
```

次の表に、このアドレス指定で使用されるパラメータを示します。

パラメータ	説明
PROTOCOL	使用するプロトコルを表します。この値は TCPS です。大/小文字は区別されません。

パラメータ	説明
HOST	ホスト名またはホストの IP アドレスを表します。
PORT	Secure Sockets Layer 付き TCP/IP ポートを表します。このポートには、番号または /etc/services ファイルでこのポートにマップされた別名を指定します。推奨値は 2484 です。

次に、Secure Sockets Layer付きTCP/IPプロトコル・アドレスのサンプルを示します。

```
(ADDRESS= (PROTOCOL=TCPS)(HOST=MADRID)(PORT=2484))
```

5.4 TCP/IPまたはSecure Sockets Layer付きTCP/IP用のリスナーの設定

リスナー用のポートは、ネットワーク上の各Oracle Net Servicesノードの/etc/servicesファイルで予約することをお薦めします。デフォルトのポートは1521です。エントリにはリスナー名とポート番号がリストされます。たとえば:

```
oraclelistener    1521/tcp
```

この例のoraclelistenerは、listener.oraファイルに定義されているリスナーの名前です。複数のリスナーを起動する場合は、複数のポートを予約してください。

Secure Sockets Layerを使用する場合は、Secure Sockets Layer付きTCP/IP用のポートを/etc/servicesファイルに定義する必要があります。推奨値は2484です。たとえば:

```
oraclelistenerssl     2484/tcps
```

この例のoraclelistenersslは、listener.oraファイルに定義されているリスナーの名前です。複数のリスナーを起動する場合は、複数のポートを予約してください。

6 OracleプリコンパイラおよびOracle Call Interfaceの使用

この章では、OracleプリコンパイラおよびOracle Call Interfaceの使用方法について説明します

トピック:

- Oracleプリコンパイラの概要
- クライアント・アプリケーションのビット長サポート
- Pro*C/C++プリコンパイラ
- Pro*COBOLプリコンパイラ
- Pro*FORTRANプリコンパイラ
- SQL*Module for ADA
- OCIとOCCI
- 64ビット・ドライバでのOracle JDBC/OCIプログラムの実行
- カスタムMakeファイル
- 未定義シンボルの修正
- マルチスレッド・アプリケーション
- シグナル・ハンドラの使用
- XA機能

6.1 Oracleプリコンパイラの概要

Oracleプリコンパイラは、Oracle DatabaseのSQL文を高水準言語で記述されたプログラムと組み合せるためのアプリケーション開発ツールです。Oracleプリコンパイラは、ANSI SQLと互換性があり、Oracle Databaseやその他のANSI SQLデータベース管理システムで実行するオープンでカスタマイズされたアプリケーションを開発するために使用します。

この項では、次の項目について説明します。

- プリコンパイラ構成ファイル
- プリコンパイラ実行可能ファイルの再リンク
- すべてのプリコンパイラに共通の問題
- 静的および動的リンク
- クライアント共有ライブラリとクライアント静的ライブラリ

ノート:

- この項で示す ORACLE_HOME は、「管理者」インストール・タイプを使用して Oracle Database

Client 19c をインストールするときに作成される ORACLE_HOME を指します。

- 読取り専用 Oracle ホームを構成している場合は、デモ・ディレクトリを ORACLE_HOME から ORACLE_BASE_HOME にコピーする必要があります。詳細は、『Oracle Database インストレーション・ガイド for Linux』を参照してください。

6.1.1 プリコンパイラ構成ファイル

Oracleプリコンパイラの構成ファイルは、`$ORACLE_HOME/precomp/admin`ディレクトリにあります。

次の表に、各プリコンパイラの構成ファイルの名前を示します。

表6-1 Oracleプリコンパイラのシステム構成ファイル

製品	構成ファイル
Pro*C/C++	`pcscfg.cfg`
Pro*COBOL	`pcbcfg.cfg`
Pro*FORTRAN (IBM AIX on POWER Systems (64-Bit)、HP-UX および Oracle Solaris)	`pccfor.cfg`
Object Type Translator	`ottcfg.cfg`
SQL*Module for Ada (IBM AIX on POWER Systems (64-Bit))	`pmscfg.cfg`

6.1.2 プリコンパイラ実行可能ファイルの再リンク

すべてのプリコンパイラ実行可能ファイルを再リンクするには、Makeファイル `$ORACLE_HOME/precomp/lib/ins_precomp.mk`を使用します。特定のプリコンパイラ実行可能ファイルを手動で再リンクするには、次のコマンドを入力します。

```
$ make -f ins_precomp.mk relink exename = executable_name
```

このコマンドを実行すると、最初に、新しい実行可能ファイルが`$ORACLE_HOME/precomp/lib`ディレクトリに作成され、次にそのファイルが`$ORACLE_HOME/bin`ディレクトリに移動されます。

この例では、`executable`を表6-2で示した製品の実行可能ファイルで置き換えます。

次の表に、Oracleプリコンパイラの実行可能ファイルを示します。

表6-2 Oracleプリコンパイラの実行可能ファイル

製品	実行可能ファイル
Pro*FORTRAN 32 ビット(Oracle Solaris、HP-UX および IBM AIX on POWER Systems (64-Bit))	`profor`

製品	実行可能ファイル
Pro*COBOL 32 ビット(Oracle Solaris、HP-UX および IBM AIX on POWER Systems (64-Bit))	`procob`
Pro*COBOL (Oracle Solaris、HP-UX および IBM AIX on POWER Systems (64-Bit))	`procob` または `rtsora`
Pro*C/C++ 32 ビット(HP-UX)	`proc`
Pro*FORTRAN (HP-UX)	`profor`
SQL*Module for Ada (IBM AIX on POWER Systems (64-Bit))	`modada`

6.1.3 すべてのプリコンパイラに共通の問題

次の問題は、すべてのプリコンパイラに共通しています。

- 大文字から小文字への変換

  C言語以外では、コンパイラによって、大文字の関数やサブプログラム名が小文字に変換されます。これが、`No such user exists`エラー・メッセージの原因になる可能性があります。このエラー・メッセージが表示された場合は、オプション・ファイル内の関数またはサブプログラム名の大/小文字が、IAPXTB表の文字と一致しているかどうかを確認してください。

- ベンダー提供のデバッガ・プログラム

  プリコンパイラとベンダー提供のデバッガに互換性がない場合があります。デバッガを使用して動作したプログラムが、デバッガを使用しないときも同様に動作するとはかぎりません。

- `IRECLEN`および`ORECLEN`の各パラメータの値

  `IRECLEN`および`ORECLEN`の各パラメータには、最大値がありません。

6.1.4 静的および動的リンク

Oracleライブラリは、プリコンパイラやOCIまたはOCCIアプリケーションと静的または動的にリンクできます。静的リンクの場合、アプリケーション全体のライブラリおよびオブジェクトは、1つの実行可能プログラムにリンクされます。その結果、アプリケーションの実行可能ファイルが非常に大きくなることがあります。

動的リンクの場合、実行コードの一部分が実行可能プログラムに格納され、残りの部分はアプリケーションの実行時に動的にリンクされるライブラリに格納されます。実行時にリンクされるライブラリを、動的ライブラリまたは共有ライブラリと呼びます。動的リンクには、次のようないくつかのメリットがあります。

- ディスク領域要件が少なくなります。複数のアプリケーションで、または同一アプリケーションのコールで、同じ動的ライブラリを使用できます。
- メイン・メモリー要件が少なくなります。同一の動的ライブラリ・イメージをメイン・メモリーに一度ロードしておくと、複数のアプリケーションでそれを共有できます。

6.1.5 クライアント共有ライブラリとクライアント静的ライブラリ

クライアント共有ライブラリとクライアント静的ライブラリは$ORACLE_HOME/lib.にあります。オラクル社が提供しているdemo_product.mkファイルを使用してアプリケーションをリンクする場合は、デフォルトでクライアント共有ライブラリがリンクされます。

共有ライブラリ・パスの環境変数設定にクライアント共有ライブラリを含むディレクトリが指定されていない場合、実行可能ファイルの起動時に、次のようなエラー・メッセージのいずれかが表示される場合があります。

```
Cannot load library libclntsh.a
cannot open shared library: .../libclntsh.sl.10.1
libclntsh.so.10.1: can't open file: errno=2
can't open library: .../libclntsh.dylib.10.1
Cannot map libclntsh.so
```

このエラーを防ぐには、共有ライブラリ・パスの環境変数を設定し、適切なディレクトリを指定します。次の表に、環境変数名のサンプル設定を示します。プラットフォームで32ビットおよび64ビットのアプリケーションがどちらもサポートされている場合、実行するアプリケーションに応じて、正しいディレクトリを指定してください。

プラットフォーム	環境変数	サンプル設定
Oracle Solaris (32 ビットおよび 64 ビット・アプリケーション)および Linux	LD_LIBRARY_PATH	$ORACLE_HOME/lib
IBM AIX on POWER Systems (32 ビットおよび 64 ビット・アプリケーション)	LIBPATH	$ORACLE_HOME/lib
HP-UX (32 ビット・アプリケーション)	SHLIB_PATH	$ORACLE_HOME/lib
HP-UX (64 ビット・アプリケーション)	LD_LIBRARY_PATH	$ORACLE_HOME/lib

クライアント共有ライブラリは、インストール時に自動的に作成されます。再作成する必要がある場合は、次の手順を実行します。

1. クライアント共有ライブラリを使用するすべてのクライアント・アプリケーションを終了します。この中には、SQL*PlusやOracle Recovery ManagerなどのOracle Clientアプリケーションもすべて含まれます。
2. oracleユーザーでログインし、次のコマンドを実行します。

```
$ $ORACLE_HOME/bin/genclntsh
```

非スレッドのクライアント共有ライブラリ

ノート:

この項の内容は、HP-UX システムに適用されます。

HP-UXでは、非スレッドのクライアント共有ライブラリを使用できます。ただし、このライブラリは、スレッドを使用する、またはスレッドに依存するOCIアプリケーションでは使用できません。

スレッドを使用しないアプリケーションでこのライブラリを使用するには、次のコマンドを実行して32および64ビットのOCIアプリケーションを作成します。

```
$ make -f demo_rdbms.mk build_nopthread EXE=oci02 OBJS=oci02.o
```

6.1.6 クライアントの静的ライブラリの生成

クライアントの静的ライブラリにアプリケーションをリンクする場合は、最初に静的ライブラリを生成する必要があります。

クライアントの静的ライブラリ(`libclntst12.a`)は、Oracle Databaseのインストール時に生成されません。クライアントの静的ライブラリにアプリケーションをリンクする場合は、最初に静的ライブラリを生成する必要があります。

1. ユーザーをOracleインストール所有者(`oracle`)に切り替えます。
2. `ORACLE_HOME`環境変数を設定して、Oracle Databaseのインストールに使用したOracleホーム・ディレクトリを指定します。たとえば:
   - Bourne、BashまたはKornシェル:

     ```
     $ ORACLE_HOME=/u01/app/oracle/product/19.0.0/dbhome_1
     $ export ORACLE_HOME
     ```

   - Cシェルの場合:

     ```
     % setenv ORACLE_HOME /u01/app/oracle/product/19.0.0/dbhome_1
     ```

3. 次のコマンドを入力します。

   ```
   $ $ORACLE_HOME/bin/genclntst
   ```

6.2 クライアント・アプリケーションのビット長サポート

クライアント・アプリケーション・タイプ(32ビットまたは64ビット)は次のプラットフォームでサポートされています。

- Oracle Solaris
- Linux x86-64
- IBM: Linux on System z
- IBM AIX on POWER Systems (64ビット)
- HP-UX Itanium

次の表に、32ビットおよび64ビットのクライアント共有ライブラリを示します。

プラットフォーム	32ビットのクライアント共有ライブラリ	64ビットのクライアント共有ライブラリ
Oracle Solaris、Linux x86-64 および IBM: Linux on System z	`$ORACLE_HOME/lib/libclntsh.so`	`$ORACLE_HOME/lib/libclntsh.so`
IBM AIX on POWER Systems (64 ビット)	`$ORACLE_HOME/lib/libclntsh.a` `$ORACLE_HOME/lib/libclntsh.so`	`$ORACLE_HOME/lib/libclntsh.a` `$ORACLE_HOME/lib/libclnts`

プラットフォーム	32ビットのクライアント共有ライブラリ	64ビットのクライアント共有ライブラリ
		h.so
HP-UX Itanium	$ORACLE_HOME/lib/libclntsh.sl	$ORACLE_HOME/lib/libclntsh.sl

異なるワード・サイズが混在しているインストールを実装するには:

1. 次のコマンドを実行して、32ビットおよび64ビットのクライアント共有ライブラリを作成します。

```
$ $ORACLE_HOME/bin/genclntsh
```

2. プラットフォームに応じて、次のいずれかの環境変数に、必要な32ビットおよび64ビットのクライアント共有ライブラリのパスを入力します。

プラットフォーム	環境変数
Oracle Solaris、Linux x86-64、IBM: Linux on System z および HP-UX	LD_LIBRARY_PATH
IBM AIX on POWER Systems (64 ビット)	LIBPATH
HP-UX(32 ビット・クライアント・アプリケーション)	SHLIB_PATH

32ビットのPro*CおよびOCIカスタマ・アプリケーションの作成

32ビットおよび64ビットのPro*CおよびOracle Call Interface(OCI)カスタマ・アプリケーションをサポートするオペレーティング・システムでは、次のファイルに32ビットPro*CおよびOCIアプリケーションの作成方法が記載されています。

参照内容	参照先のMakeファイル . .
32 ビットの Pro*C アプリケーションの作成	$ORACLE_HOME/precomp/demo/proc/demo_proc.mk
32 ビットの OCI アプリケーションの作成	$ORACLE_HOME/rdbms/demo/demo_rdbms.mk

6.3 Pro*C/C++プリコンパイラ

Pro*C/C++プリコンパイラを使用する場合は、事前にオペレーティング・システムの適切なバージョンのコンパイラが正しくインストールされていることを確認してください。

関連項目:

- サポートされるコンパイラのバージョンの詳細は、『Oracle Databaseインストレーション・ガイド』を参照してください。

- Pro*C/C++プリコンパイラおよびインタフェース機能の詳細は、『Pro*C/C++プログラマーズ・ガイド』を参照してください。

この項では、次の項目について説明します。

- Pro*C/C++のデモ・プログラム
- Pro*C/C++のユーザー・プログラム

6.3.1 Pro*C/C++のデモ・プログラム

デモ・プログラムは、Pro*C/C++プリコンパイラの機能を紹介するために用意されています。デモ・プログラムには、C、C++およびObjectプログラムの3種類があります。デモ・プログラムはすべて、`$ORACLE_HOME/precomp/demo/proc`ディレクトリにあります。デフォルトでは、すべてのプログラムがクライアント共有ライブラリに動的にリンクされます。

デモ・プログラムを実行するには、`demobld.sql`スクリプトで作成したデモンストレーション表がパスワード付きのJONESスキーマに存在する必要があります。

`demobld.sql`スクリプトは、読取り/書込みOracleホームの`$ORACLE_HOME/sqlplus/demo/`ディレクトリと、読取り専用のOracleホームの`$(orabasehome)/sqlplus/demo/`ディレクトリにあります。

ノート:

デモンストレーションを作成する前に、JONES アカウントのロックを解除してパスワードを設定する必要があります。

次の`demo_proc.mk` Makeファイルを使用してデモ・プログラムを作成します。`demo_proc.mk`ファイルは、読取り/書込みOracleホームの`$ORACLE_HOME/precomp/demo/proc/`ディレクトリと、読取り専用のOracleホームの`$(orabasehome/precomp/demo/proc/`にあります。たとえば、`sample1`のデモ・プログラムをプリコンパイル、コンパイルおよびリンクするには、次のコマンドを実行します。

```
$ make -f demo_proc.mk sample1
```

ノート:

IBM AIX on POWER Systems (64-Bit)の場合、デモ・プログラムを正しくコンパイルするために、次の例の`make`コマンドの`-r`オプションを指定してください。たとえば:

```
$ make -r -f demo_proc.mk sample1
```

Pro*C/C++のCデモ・プログラムをすべて作成するには、次のコマンドを実行します。

```
$ make -f demo_proc.mk samples
```

Pro*C/C++のC++デモ・プログラムをすべて作成するには、次のコマンドを実行します。

```
$ make -f demo_proc.mk cppsamples
```

Pro*C/C++のObjectデモ・プログラムをすべて作成するには、次のコマンドを実行します。

```
$ make -f demo_proc.mk object_samples
```

一部のデモ・プログラムでは、$ORACLE_HOME/precomp/demo/sqlディレクトリ内にあるSQLスクリプトを実行する必要があります。このスクリプトを実行しないと、実行を要求するメッセージが表示されます。

デモ・プログラムを作成し、それに対応するSQLスクリプトを実行するには、「makeマクロ引数RUNSQL=run」をコマンドラインに追加します。たとえば、sample9デモ・プログラムを作成し、必要な$ORACLE_HOME/precomp/demo/sql/sample9.sqlスクリプトを実行するには、次のコマンドを実行します。

```
$ make -f demo_proc.mk sample9 RUNSQL=run
```

Objectデモ・プログラムのすべてを作成し、必要なSQLスクリプトすべてを実行するには、次のコマンドを実行します。

```
$ make -f demo_proc.mk object_samples RUNSQL=run
```

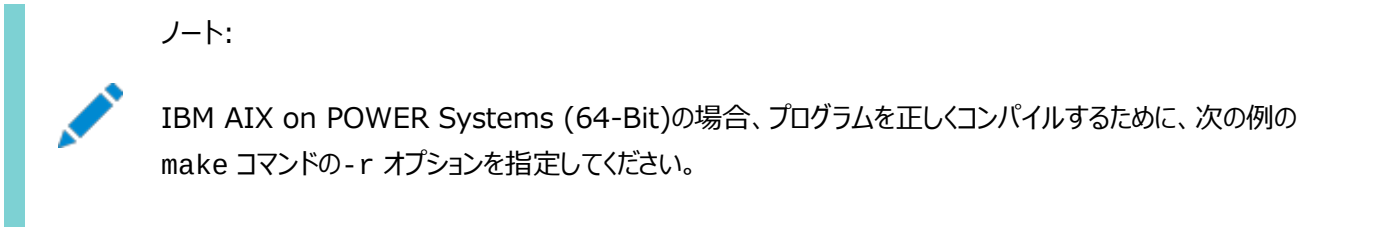
6.3.2 Pro*C/C++のユーザー・プログラム

Makeファイル$ORACLE_HOME/precomp/demo/proc/demo_proc.mkを使用して、ユーザー・プログラムを作成できます。このMakeファイルでは、32ビットおよび64ビットの両方のユーザー・プログラムが作成されます。次の表では、Pro*C/C++でユーザー・プログラムを作成するためのMakeファイルを示します。

プラットフォーム	Makeファイル
Oracle Solaris、Linux x86-64、IBM: Linux on System z、IBM AIX on POWER Systems (64-Bit)および HP-UX	demo_proc.mk

関連項目:

ユーザー・プログラムの作成方法は、そのMakeファイルを参照してください。



ノート:

IBM AIX on POWER Systems (64-Bit)の場合、プログラムを正しくコンパイルするために、次の例のmake コマンドの-r オプションを指定してください。

Makeファイルdemo_proc.mkを使用してプログラムを作成する場合は、次のようなコマンドを実行します。

```
$ make -f demo_proc.mk target OBJS="objfile1 objfile2 ..." EXE=exename
```

この例では、次のようになります。

- targetは、使用するMakeファイルのターゲットです。
- objfilenは、プログラムをリンクするためのオブジェクト・ファイルです。
- exenameは、実行可能プログラムです。

たとえば、Pro*C/C++ソース・ファイルmyprog.pcからプログラムmyprogを作成する場合は、作成する実行可能ファイルのソースとタイプに応じて、次のいずれかのコマンドを実行します。

- Cソースの場合、クライアント共有ライブラリに動的にリンクさせるには、次のコマンドを実行します。

  ```
  $ make -f demo_proc.mk build OBJS=myprog.o EXE=myprog
  ```

- Cソースの場合、クライアント共有ライブラリに静的にリンクさせるには、次のコマンドを実行します。

```
$ make -f demo_proc.mk build_static OBJS=myprog.o EXE=myprog
```

- C++ソースの場合、クライアント共有ライブラリに動的にリンクさせるには、次のコマンドを実行します。

```
$ make -f demo_proc.mk cppbuild OBJS=myprog.o EXE=myprog
```

- C++ソースの場合、クライアント共有ライブラリに静的にリンクさせるには、次のコマンドを実行します。

```
$ make -f demo_proc.mk cppbuild_static OBJS=myprog.o EXE=myprog
```

6.4 Pro*COBOLプリコンパイラ

表6-3に、Pro*COBOLプリコンパイラのネーミング規則を示します。

表6-3 Pro*COBOLネーミング規則

アイテム	ネーミング規則
実行可能ファイル	`procob`
デモンストレーション・ディレクトリ	`procob2`
Make ファイル	`demo_procob.mk`

Pro*COBOLでは、静的リンク、動的リンクまたは動的読取りプログラムをサポートしています。動的リンク・プログラムは、クライアント共有ライブラリを使用します。動的読取りプログラムは、`$ORACLE_HOME/bin`ディレクトリにある`rtsora`実行可能ファイルを使用します。

この項では、次の項目について説明します。

- Pro*COBOLの環境変数
- Pro*COBOLのOracleランタイム・システム
- Pro*COBOLのデモ・プログラム
- Pro*COBOLのユーザー・プログラム
- FORMATプリコンパイラ・オプション

6.4.1 Pro*COBOLの環境変数

この項では、Pro*COBOLで必要な環境変数について説明します。

- Micro Focus Server Express COBOLコンパイラ
- Acucorp ACUCOBOL-GT COBOLコンパイラ

6.4.1.1 Micro Focus Server Express COBOLコンパイラ

Micro Focus Server Express COBOLコンパイラを使用するには、環境変数`COBDIR`と`PATH`および共有ライブラリ・パス環境変数を設定する必要があります。

関連項目:

共有ライブラリのパス環境変数の詳細は、「クライアント共有ライブラリとクライアント静的ライブラリ」を参照してください

COBDIR

環境変数COBDIRには、コンパイラがインストールされているディレクトリを設定します。たとえば、コンパイラが/opt/lib/cobolディレクトリにインストールされている場合は、次のコマンドを実行します。

- Bourne、BashまたはKornシェル:

```
$ COBDIR=/opt/lib/cobol
$ export COBDIR
```

- Cシェルの場合:

```
% setenv COBDIR /opt/lib/cobol
```

PATH

$COBDIR/binディレクトリを含むように環境変数PATHを設定します。

- Bourne、BashまたはKornシェル:

```
$ PATH=$COBDIR/bin:$PATH
$ export PATH
```

- Cシェルの場合:

```
% setenv PATH ${COBDIR}/bin:${PATH}
```

共有ライブラリ・パス

環境変数LIBPATH、LD_LIBRARY_PATHまたはSHLIB_PATHには、コンパイラ・ライブラリがインストールされているディレクトリを設定します。たとえば、プラットフォームで環境変数LD_LIBRARY_PATHが使用されており、コンパイラ・ライブラリが$COBDIR/coblibディレクトリにインストールされている場合は、次のコマンドを実行します。

- Bourne、BashまたはKornシェル:

```
$ LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$COBDIR/coblib
$ export LD_LIBRARY_PATH
```

- Cシェルの場合:

```
% setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:$COBDIR/coblib
```

6.4.1.2 Acucorp ACUCOBOL-GT COBOLコンパイラ

Acucorp ACUCOBOL-GT COBOLコンパイラを使用するには、A_TERMCAP、A_TERM、PATHおよびLD_LIBRARY_PATH環境変数を設定する必要があります。環境変数LD_LIBRARY_PATHに正しいディレクトリが設定されていない場合は、プログラムのコンパイルまたは実行時に、次のようなエラー・メッセージが表示されます。

```
runcbl: error while loading shared libraries: libclntsh.so:
cannot open shared object file: No such file or directory
```

A_TERMCAPおよびA_TERM

a_termcapファイルの場所を指定するには環境変数A_TERMCAPを設定し、そのファイルからサポートされる端末を指定する

には環境変数A_TERMを設定します。たとえば:

- Bourne、BashまたはKornシェル:

```
$ A_TERMCAP=/opt/COBOL/etc/a_termcap
$ A_TERM=vt100
$ export A_TERMCAP A_TERM
```

- Cシェルの場合:

```
% setenv A_TERMCAP /opt/COBOL/etc/a_termcap
% setenv A_TERM vt100
```

PATH

/opt/COBOL/binディレクトリを含むように環境変数PATHを設定します。

- Bourne、BashまたはKornシェル:

```
$ PATH=/opt/COBOL/bin:$PATH
$ export PATH
```

- Cシェルの場合:

```
% setenv PATH opt/COBOL/bin:${PATH}
```

LD_LIBRARY_PATH

ノート:

IBM AIX on POWER Systems (64-Bit)の場合、LIBPATH 変数は LD_LIBRARY_PATH 変数と同等です。IBM AIX on POWER Systems (64-Bit)では、次のコマンドで LD_LIBRARY_PATH 変数ではなく LIBPATH 変数を使用する必要があります。

環境変数LD_LIBRARY_PATHには、コンパイラ・ライブラリがインストールされているディレクトリを設定します。たとえば、コンパイラ・ライブラリが/opt/COBOL/libディレクトリにインストールされている場合は、次のコマンドを実行します。

- Bourne、BashまたはKornシェル:

```
$ LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/COBOL/lib
$ export LD_LIBRARY_PATH
```

- Cシェルの場合:

```
% setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/opt/COBOL/lib
```

6.4.2 Pro*COBOLのOracleランタイム・システム

Oracleには、動的読取りPro*COBOLプログラムを実行するために、rtsoraという専用のランタイム・システムが用意されています。動的読取りPro*COBOLプログラムを実行するには、cobrunランタイム・システムのかわりに、rtsoraランタイム・システムを使用します。cobrunを使用してPro*COBOLプログラムを実行すると、次のようなエラー・メッセージが表示されます。

```
$ cobrun sample1.gnt
Load error : file 'SQLADR'
error code: 173, pc=0, call=1, seg=0
173     Called program file not found in drive/directory
```

6.4.3 Pro*COBOLのデモ・プログラム

デモ・プログラムは、Pro*COBOLプリコンパイラの機能を紹介するために用意されています。デモ・プログラムは、$ORACLE_HOME/precomp/demo/procob2ディレクトリにあります。デフォルトでは、すべてのプログラムがクライアント共有ライブラリに動的にリンクされます。

デモ・プログラムを実行するには、$ORACLE_HOME/sqlplus/demo/demobld.sqlスクリプトで作成したデモンストレーション表がパスワード付きのJONESスキーマに存在する必要があります。

ノート:

デモンストレーションを作成する前に、JONES アカウントのロックを解除してパスワードを設定する必要があります。

次のMakeファイルを使用してデモ・プログラムを作成します。

```
$ORACLE_HOME/precomp/demo/procob2/demo_procob.mk
```

sample1というPro*COBOL用のデモ・プログラムをプリコンパイル、コンパイルおよびリンクするには、次のコマンドを実行します。

```
$ make -f demo_procob.mk sample1
```

Pro*COBOLデモ・プログラムを作成するには、次のコマンドを実行します。

```
$ make -f demo_procob.mk samples
```

rtsoraランタイム・システムで使用する動的読取りプログラムsample1.gntを作成し、実行するには、次のコマンドを実行します。

```
$ make -f demo_procob.mk sample1.gnt
$ rtsora sample1.gnt
```

一部のデモ・プログラムでは、$ORACLE_HOME/precomp/demo/sqlディレクトリ内にあるSQLスクリプトを実行する必要があります。このスクリプトを実行しないと、実行を要求するメッセージが表示されます。

デモ・プログラムを作成し、それに対応するSQLスクリプトを実行するには、「makeマクロ引数RUNSQL=run」をコマンドに追加します。たとえば、sample9デモ・プログラムを作成し、必要な$ORACLE_HOME/precomp/demo/sql/sample9.sqlスクリプトを実行するには、次のコマンドを実行します。

```
$ make -f demo_procob.mk sample9 RUNSQL=run
```

Pro*COBOLデモ・プログラムを作成し、必要なSQLスクリプトすべてを実行するには、次のコマンドを実行します。

```
$ make -f demo_procob.mk samples RUNSQL=run
```

6.4.4 Pro*COBOLのユーザー・プログラム

Makeファイル$ORACLE_HOME/precomp/demo/procob2/demo_procob.mkを使用して、ユーザー・プログラムを作成できます。このMakeファイルでは、32ビットおよび64ビットの両方のユーザー・プログラムが作成されます。次の表では、Pro*COBOLでユーザー・プログラムを作成するためのMakeファイルを示します。

プラットフォーム	Makeファイル

プラットフォーム	Makeファイル
Oracle Solaris、Linux x86-64、IBM: Linux on System z、IBM AIX on POWER Systems (64-Bit)および HP-UX	`demo_procob.mk`

関連項目:

ユーザー・プログラムの作成方法は、そのMakeファイルを参照してください。

Makeファイル`demo_procob.mk`を使用してプログラムを作成する場合は、次のようなコマンドを実行します。

```
$ make -f demo_procob.mk target COBS="cobfile1 cobfile2 ..." EXE=exename
```

この例では、次のようになります。

- `target`は、使用するMakeファイルのターゲットです。
- `cobfilen`は、プログラムのCOBOLソース・ファイルです。
- `exename`は、実行可能プログラムです。

たとえば、プログラム`myprog`を作成する場合は、作成する実行可能ファイルのソースとタイプに応じて、次のいずれかのコマンドを実行します。

- COBOLソースの場合、クライアント共有ライブラリに動的にリンクさせるには、次のコマンドを実行します。

  ```
  $ make -f demo_procob.mk build COBS=myprog.cob EXE=myprog
  ```

- COBOLソースの場合、静的にリンクさせるには、次のコマンドを実行します。

  ```
  $ make -f demo_procob.mk build_static COBS=myprog.cob EXE=myprog
  ```

- COBOLソースの場合、`rtsora`で使用するための動的読取りプログラムを作成するには、次のコマンドを実行します。

  ```
  $ make -f demo_procob.mk myprog.gnt
  ```

6.4.5 FORMATプリコンパイラ・オプション

FORMATプリコンパイラ・オプションは、COBOLの入力行の形式を指定します。デフォルト値のANSIを指定した場合、列1から6はオプションの順序番号、列7はコメントまたは継続行を示します。段落名は列8から11で開始され、列12から72が文となります。

値`TERMINAL`を指定した場合、列1から6は削除され、列7が左端の列になります。

6.5 Pro*FORTRANプリコンパイラ

Pro*FORTRANプリコンパイラを使用する場合は、事前に適切なバージョンのコンパイラがインストールされていることを確認してください。この項では、次の項目について説明します。

- Pro*FORTRANのデモ・プログラム
- Pro*FORTRANのユーザー・プログラム

関連項目:

- サポートされるコンパイラのバージョンの詳細は、『Oracle Databaseインストレーション・ガイド』を参照してください。
- Pro*FORTRANプリコンパイラおよびインタフェース機能の詳細は、『Pro*FORTRAN Supplement to the Oracle Precompilers Guide』を参照してください。

6.5.1 Pro*FORTRANのデモ・プログラム

デモ・プログラムは、Pro*FORTRANプリコンパイラの機能を紹介するために用意されています。デモ・プログラムはすべて、`$ORACLE_HOME/precomp/demo/profor`ディレクトリにあります。デフォルトでは、すべてのプログラムがクライアント共有ライブラリに動的にリンクされます。

デモ・プログラムを実行するには、`$ORACLE_HOME/sqlplus/demo/demobld.sql`スクリプトで作成したデモンストレーション表がパスワード付きのJONESスキーマに存在する必要があります。

ノート:

デモンストレーションを作成する前に、JONES アカウントのロックを解除してパスワードを設定する必要があります。

デモ・プログラムを作成するには、`$ORACLE_HOME/precomp/demo/profor`ディレクトリにあるMakeファイル`demo_profor.mk`を使用します。たとえば、`sample1`のデモ・プログラムをプリコンパイル、コンパイルおよびリンクするには、次のコマンドを実行します。

```
$ make -f demo_profor.mk sample1
```

Pro*FORTRANのデモ・プログラムを作成するには、次のコマンドを実行します。

```
$ make -f demo_profor.mk samples
```

一部のデモ・プログラムでは、`$ORACLE_HOME/precomp/demo/sql`ディレクトリ内にあるSQLスクリプトを実行する必要があります。このスクリプトを実行しないと、実行を要求するメッセージが表示されます。

デモ・プログラムを作成し、それに対応するSQLスクリプトを実行するには、「`make`マクロ引数`RUNSQL=run`」をコマンドラインに追加します。たとえば、`sample11`デモ・プログラムを作成し、必要な`$ORACLE_HOME/precomp/demo/sql/sample11.sql`スクリプトを実行するには、次のコマンドを実行します。

```
$ make -f demo_profor.mk sample11 RUNSQL=run
```

Pro*FORTRANのデモ・プログラムを作成し、必要なSQLスクリプトすべてを実行するには、次のコマンドを実行します。

```
$ make -f demo_profor.mk samples RUNSQL=run
```

6.5.2 Pro*FORTRANのユーザー・プログラム

Makeファイル`$ORACLE_HOME/precomp/demo/profor/demo_profor.mk`を使用して、ユーザー・プログラムを作成できます。このMakeファイルでは、32ビットおよび64ビットの両方のユーザー・プログラムが作成されます。次の表では、Pro*FORTRANでユーザー・プログラムを作成するためのMakeファイルを示します。

プラットフォーム	Makeファイル

プラットフォーム	Makeファイル
Oracle Solaris、IBM AIX on POWER Systems (64-Bit)および HP-UX	demo_profor.mk

関連項目:

ユーザー・プログラムの作成方法は、そのMakeファイルを参照してください。

Makeファイルdemo_proc.mkを使用してプログラムを作成する場合は、次のようなコマンドを実行します。

```
$ make -f demo_profor.mk target FORS="forfile1 forfile2 ..." EXE=exename
```

この例では、次のようになります。

- targetは、使用するMakeファイルのターゲットです。
- forfilenは、プログラムのFORTRANソース・ファイルです。
- exenameは、実行可能プログラムです。

たとえば、Pro*FORTRANソース・ファイルmyprog.pfoからプログラムmyprogを作成する場合は、作成する実行可能ファイルのタイプに応じて、次のいずれかのコマンドを実行します。

- 実行可能ファイルの場合、クライアント共有ライブラリに動的にリンクさせるには、次のコマンドを実行します。

  ```
  $ make -f demo_profor.mk build FORS=myprog.f EXE=myprog
  ```

- 実行可能ファイルの場合、クライアント共有ライブラリに静的にリンクさせるには、次のコマンドを実行します。

  ```
  $ make -f demo_profor.mk build_static FORS=myprog.f EXE=myprog
  ```

6.6 SQL*Module for ADA

ノート:

この項の内容は、IBM AIX on POWER Systems (64-Bit)プラットフォームに適用されます。

SQL*Module for Adaを使用する場合は、事前に適切なバージョンのコンパイラがインストールされていることを確認してください。

関連項目:

- 必要なコンパイラのバージョンの詳細は、『Oracle Databaseインストレーション・ガイド』を参照してください。
- SQL*Module for Adaの詳細は、『Oracle SQL*Module for Ada Programmer's Guide』を参照してください。

この項では、次の項目について説明します。

- SQL*Module for Adaデモ・プログラム

- SQL*Module for Adaユーザー・プログラム

6.6.1 SQL*Module for Adaデモ・プログラム

デモ・プログラムは、SQL*Module for Adaの機能を紹介するために用意されています。デモ・プログラムはすべて、$ORACLE_HOME/precomp/demo/modadaディレクトリにあります。デフォルトでは、すべてのプログラムがクライアント共有ライブラリに動的にリンクされます。

ch1_drvデモ・プログラムを実行するには、$ORACLE_HOME/sqlplus/demo/demobld.sqlスクリプトで作成したデモンストレーション表がパスワード付きのJONESスキーマに存在する必要があります。

ノート:

デモンストレーションを作成する前に、JONES アカウントのロックを解除してパスワードを設定する必要があります。

demcalspおよびdemohostの各デモ・プログラムでは、サンプルのcollegeデータベースがMODTESTスキーマに存在している必要があります。適切なmakeコマンドを使用してMODTESTスキーマを作成すると、サンプルのcollegeデータベースをロードできます。

SQL*Module for Adaデモ・プログラムを作成して、MODTESTユーザーの作成に必要なSQLスクリプトを実行し、サンプルのcollegeデータベースを作成するには、次のコマンドを実行します。

```
$ make -f demo_modada.mk all RUNSQL=run
```

1つのデモ・プログラム(demohost)を作成して、MODTESTユーザーの作成に必要なSQLスクリプトを実行し、サンプルのcollegeデータベースを作成するには、次のコマンドを実行します。

```
$ make -f demo_modada.mk makeuser loaddb demohost RUNSQL=run
```

SQL*Module for Adaデモ・プログラムを作成して、サンプルのcollegeデータベースを再び作成しない場合には、次のコマンドを実行します。

```
$ make -f demo_modada.mk samples
```

1つのデモ・プログラム(demohost)を作成して、サンプルのcollegeデータベースを再び作成しない場合は、次のコマンドを実行します。

```
$ make -f demo_modada.mk demohost
```

プログラムを実行するには、Oracle Netの接続文字列を定義するか、または適切な表が存在するデータベースへの接続に使用するINST1_ALIASという別名を定義する必要があります。

6.6.2 SQL*Module for Adaユーザー・プログラム

Makeファイル$ORACLE_HOME/precomp/demo/modada/demo_modada.mkを使用して、ユーザー・プログラムを作成できます。Makeファイルdemo_modada.mkを使用してプログラムを作成する場合は、次のようなコマンドを実行します。

```
$ make -f demo_modada.mk ada OBJS="module1 module2 ..." ¥
EXE=exename MODARGS=SQL_Module_arguments
```

この例では、次のようになります。

- `modulen`は、コンパイル済のAdaオブジェクトです。
- `exename`は、実行可能プログラムです。
- `SQL_Module_arguments`は、SQL*Moduleに渡されるコマンドライン引数です。

関連項目:

SQL*Module for Adaの詳細は、『Oracle SQL*Module for Ada Programmer's Guide』を参照してください。

6.7 OCIおよびOCCI

Oracle Call Interface(OCI)またはOracle C++ Call Interface(OCCI)を使用する前に、適切なバージョンのCまたはC++がインストール済であることを確認してください。

関連項目:

- サポートされるコンパイラのバージョンの詳細は、『Oracle Databaseインストレーション・ガイド』を参照してください。
- OCIおよびOCCIについては、Oracle Call Interfaceのプログラマーズ・ガイドまたはOracle C++ Call Interfaceのプログラマーズ・ガイドを参照してください。

この項では、次の項目について説明します。

- OCIとOCCIのデモ・プログラム
- OCIとOCCIのユーザー・プログラム

6.7.1 OCIとOCCIのデモ・プログラム

OCIとOCCIの機能を紹介するデモ・プログラムは、Oracle Database 19c Examplesソフトウェアに収録されています。デモ・プログラムには、CおよびC++プログラムの2種類があります。デモ・プログラムは、`$ORACLE_HOME/rdbms/demo`ディレクトリにあります。デフォルトでは、すべてのプログラムがクライアント共有ライブラリに動的にリンクされます。

デモ・プログラムを実行するには、`$ORACLE_HOME/sqlplus/demo/demobld.sql`スクリプトで作成したデモンストレーション表がパスワード付きのJONESスキーマに存在する必要があります。一部のデモ・プログラムでは、デモ・ソース・ファイルに示されているように、特定の.sqlファイルを実行することが必要です。OCCIデモ・プログラムでは、occidemo.sqlを実行することが必要です。

ノート:

デモンストレーションを作成する前に、JONES アカウントのロックを解除してパスワードを設定する必要があります。

デモ・プログラムを作成するには、`$ORACLE_HOME/rdbms/demo`ディレクトリにあるMakeファイル`demo_rdbms.mk`を使用します。たとえば、`cdemo1`のデモ・プログラムをコンパイルおよびリンクするには、次のコマンドを実行します。

```
$ make -f demo_rdbms.mk cdemo1
```

OCIのCデモ・プログラムを作成するには、次のコマンドを実行します。

```
$ make -f demo_rdbms.mk demos
```

OCCIのC++デモ・プログラムを作成するには、次のコマンドを実行します。

```
$ make -f demo_rdbms.mk occidemos
```

6.7.2 OCIとOCCIのユーザー・プログラム

Makeファイル$ORACLE_HOME/rdbms/demo/demo_rdbms.mkを使用して、ユーザー・プログラムを作成できます。このMakeファイルは、32ビットまたは64ビットのユーザー・プログラムを作成します。次の表に、Pro*FORTRANにより32ビットおよび64ビットのユーザー・プログラムを作成するためのMakeファイルを示します。

プラットフォーム	Makeファイル
Oracle Solaris、Linux x86-64、IBM AIX on POWER Systems (64-Bit)および HP-UX	demo_rdbms.mk

関連項目:

ユーザー・プログラムの作成方法は、そのMakeファイルを参照してください。

6.8 64ビット・ドライバでのOracle JDBC/OCIプログラムの実行

ノート:

- この項の情報は、Oracle Solaris、Linux x86-64、IBM: Linux on System z、IBM AIX on POWER Systems (64-Bit)および HP-UX の各プラットフォームに当てはまります。
- この項で説明する指示と Make ファイルを使用して、64 ビット・ドライバを使用する JDBC/OCI ユーザー・プログラムを作成できます。

64ビット・ドライバを使用するJDBC/OCIデモ・プログラムを実行するには:

1. 次の各ファイルについて、環境変数CLASSPATHの先頭に$ORACLE_HOME/jdbc/lib/ojdbc5.jarを追加します。

   ```
   jdbc/demo/samples/jdbcoci/Makefile
   jdbc/demo/samples/generic/Inheritance/Inheritance1/Makefile
   jdbc/demo/samples/generic/Inheritance/Inheritance2/Makefile
   jdbc/demo/samples/generic/Inheritance/Inheritance3/Makefile
   jdbc/demo/samples/generic/JavaObject1/Makefile
   jdbc/demo/samples/generic/NestedCollection/Makefile
   ```

2. $ORACLE_HOME/jdbc/demo/samples/generic/Makefileファイル内で、JAVA変数とJAVAC変数を変更し、JDKの位置および-d64フラグを次のように指定します。

   ```
   JAVA=${ORACLE_HOME}/java/bin/java -d64
   JAVAC=${ORACLE_HOME}/java/bin/javac -d64
   ```

3. $ORACLE_HOME/libディレクトリを含むように、環境変数LD_LIBRARY_PATH_64を設定します。

> ノート:
>
> IBM AIX on POWER Systems (64-Bit)の場合、LIBPATH 変数は LD_LIBRARY_PATH_64 変数と同等です。IBM AIX on POWER Systems (64-Bit)では、LD_LIBRARY_PATH_64 変数ではなく LIBPATH 変数を使用する必要があります。

6.9 カスタムMakeファイル

この章の製品別の項で説明したように、ユーザー・プログラムを作成する場合は、オラクル社がソフトウェアとともに提供するMakeファイルdemo_product.mkを使用してください。このMakeファイルを変更する場合、またはカスタムMakeファイルを使用する場合は、次の制限事項に注意してください。

- Oracleライブラリの順番は変更しないでください。リンク中にすべてのシンボルが解決されるように、Oracleライブラリはリンク・ラインに2回以上追加されます。

  IBM AIX on POWER Systems (64-Bit)を除き、Oracleライブラリの順序がすべてのプラットフォームで重要である理由は、次のとおりです。

  - Oracleライブラリは相互に参照し合います。たとえば、ライブラリAの関数はライブラリBの関数をコールし、ライブラリBの関数はライブラリAの関数をコールします。
  - HP-UXのリンカーは、1パス・リンカーです。IBM AIX on POWER Systems (64-Bit)、LinuxおよびOracle Solarisのリンカーは、2パス・リンカーです。
- ライブラリをリンク・ラインの最初または最後に追加します。Oracleライブラリ間にユーザー・ライブラリを入れないでください。
- nmakeまたはGNU makeなどのmakeユーティリティを使用する場合は、マクロおよび接頭辞の処理について、オペレーティング・システムで提供されているmakeユーティリティとの違いに注意してください。OracleのMakeファイルは、makeユーティリティによってテストおよびサポートされます。
- Oracleライブラリの名前および内容は、リリース間で変更されることがあります。必要なライブラリを判断するには、現行のリリースで提供されているMakeファイルdemo_product.mkを必ず使用してください。

6.10 未定義シンボルの修正

オラクル社が提供しているsymfindユーティリティを使用すると、シンボルが定義されているライブラリまたはオブジェクト・ファイルの場所を確認する際に役立ちます。プログラムのリンク時に、未定義シンボルは一般的なエラーの1つとみなされ、次のようなエラー・メッセージが生成されます。

```
$ make -f demo_proc.mk sample1
Undefined                       first referenced
 symbol                             in file
sqlcex                              sample1.o
sqlglm                              sample1.o
ld: irrecoverable: Symbol referencing errors. No output written to sample1
```

このエラーは、参照するシンボルの定義をリンカーが検出できなかった場合に発生します。このエラー・メッセージが表示された場合は、シンボルが定義されているライブラリまたはオブジェクト・ファイルがリンク・ラインにあるかどうか、およびリンカーが検索している

ファイルのディレクトリが正しいかどうかを確認します。

次の例は、symfindユーティリティの出力です。このユーティリティをsqlcexシンボルの検索に使用しています。

```
$ symfind sqlcex
SymFind - Find Symbol <sqlcex> in <**>.a, .o, .so
-----------------------------------------------------
Command:         /u01/app/oracle/product/19.0.0/bin/symfind sqlcex
Local Directory: /u01/app/oracle/product/19.0.0
Output File:     (none)
Note:            I do not traverse symbolic links
                 Use '-v' option to show any symbolic links
Locating Archive and Object files ...
[11645] |    467572|      44|FUNC |GLOB |0    |8      |sqlcex
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ./lib/libclntsh.sl
[35]    |         0|      44|FUNC |GLOB |0    |5      |sqlcex
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ./lib/libsql.a
```

6.11 マルチスレッド・アプリケーション

このリリースで提供されているOracleライブラリはスレッド・セーフで、マルチスレッド・アプリケーションをサポートします。

関連項目:

マルチスレッド・アプリケーションの詳細は、『Pro*C/C++プログラマーズ・ガイド』を参照してください。

6.12 シグナル・ハンドラの使用

Oracle Databaseでは、2タスク通信にいくつかのシグナルが使用されます。シグナルは、プロセスがデータベースに接続するとユーザー・プロセスにインストールされ、切断すると削除されます。

次の表では、Oracle Databaseで2タスク通信に使用されるシグナルについて説明します。

表6-4 2タスク通信に使用されるシグナル

シグナル	説明
SIGCLD	Oracle プロセスが終了すると、パイプ・ドライバは SIGCLD(SIGCHLD ともいいます)を使用します。SIGCLD シグナルは、オペレーティング・システム・カーネルによってユーザー・プロセスに送信されます。シグナル・ハンドラは、wait( )ルーチンを使用して、サーバー・プロセスが異常終了しているかどうかを調べます。Oracle プロセスではなく、ユーザー・プロセスが SIGCLD を受け取ります。
SIGCONT	パイプ 2 タスク・ドライバが、バンド外ブレークをユーザー・プロセスから Oracle プロセスに送信する場合に、SIGCONT を使用します。
SIGINT	2 タスク・ドライバが、ユーザーの割込み要求を検出する場合に、SIGINT を使用します。Oracle プロセスではなく、ユーザー・プロセスが SIGINT を受け取ります。
SIGIO	Oracle Net プロトコルが、ネットワーク・イベントの着信を示す場合に、SIGIO を使用します。

シグナル	説明
SIGPIPE	パイプ・ドライバが、通信チャネルのファイルの終わりを検出する場合に、SIGPIPE を使用します。パイプへの書込み時に読取りプロセスが存在していない場合、SIGPIPE シグナルが書込みプロセスに送信されます。Oracle プロセスとユーザー・プロセスの両方が、SIGPIPE を受け取ります。SIGCLD は、SIGPIPE に似ていますが、ユーザー・プロセスのみに適用され、Oracle プロセスには適用されません。
SIGTERM	パイプ・ドライバが、ユーザー側から Oracle プロセスに割込みシグナルを送る場合に、SIGTERM を使用します。ユーザーが割込みキー([Ctrl]+[C])を押すと、このシグナルが送信されます。ユーザー・プロセスではなく、Oracle プロセスが SIGTERM を受け取ります。
SIGURG	Oracle Net TCP/IP ドライバが、バンド外ブレークをユーザー・プロセスから Oracle プロセスに送信する場合に、SIGURG を使用します。

表に記載されているシグナルは、すべてのプリコンパイラ・アプリケーションに影響します。Oracleプロセスへの接続時に、SIGCLD(またはSIGCHLD)およびSIGPIPEにシグナル・ハンドラを1つのみインストールできます。`osnsui( )`ルーチンをコールして設定すると、複数のシグナル・ハンドラをSIGINT用にインストールできます。SIGINTの場合は、`osnsui( )`および`osncui( )`を使用して、シグナル受取りルーチンを登録および削除します。

また、必要に応じて、他のシグナルにもシグナル・ハンドラをインストールできます。 Oracleプロセスに接続していない場合は、複数のシグナル・ハンドラをインストールできます。

例6-1に、シグナル・ルーチンおよび受取りルーチンの設定方法を示します。

例6-1 シグナル・ルーチンおよび受取りルーチン

```
/* user side interrupt set */
word osnsui( /*_ word *handlp, void (*astp), char * ctx, _*/)
/*
** osnsui: Operating System dependent Network Set User-side Interrupt. Add an
** interrupt handling procedure astp. Whenever a user interrupt(such as a ^C)
** occurs, call astp with argument ctx. Put in *handlp handle for this
** handler so that it may be cleared with osncui. Note that there may be many
** handlers; each should be cleared using osncui. An error code is returned if
** an error occurs.
*/
/* user side interrupt clear */
word osncui( /*_ word handle _*/ );
/*
** osncui: Operating System dependent Clear User-side Interrupt. Clear the
** specified handler. The argument is the handle obtained from osnsui. An error
** code is returned if an error occurs.
*/
```

例6-2に、アプリケーション・プログラムでの`osnsui( )`および`osncui( )`ルーチンの使用方法を示します。

例6-2 osnsui()およびosncui()ルーチンのテンプレート

```
/*
** User interrupt handler template.
*/
void sig_handler()
{
...
}
main(argc, argv)
```

```
int arc;
char **argv;
{
        int handle, err;
        ...
        /* Set up the user interrupt handler */
        if (err = osnsui(&handle, sig_handler, (char *) 0))
        {
                /* If the return value is nonzero, then an error has occurred
                   Take appropriate action for the error. */
                ...
        }
        ...
        /* Clear the interrupt handler */
        if (err = osncui(handle))
        {
                /* If the return value is nonzero, then an error has occurred
                   Take appropriate action for the error. */
                ...
        }
        ...
}
```

6.13 XA機能

Oracle XAは、X/Open Distributed Transaction Processing XAインタフェースのOracle実装です。XA標準には、トランザクション内の共有リソースへのアクセスを制御するリソース・マネージャ間や、トランザクションを監視および解決するトランザクション・サービス間の双方向インタフェースが規定されています。

Oracle Call Interfaceには、XA機能があります。TPモニターXAアプリケーションを作成するときは、TPモニター・ライブラリ(シンボルax_regおよびax_unregを定義するライブラリ)が、リンク・ラインでOracle Client共有ライブラリより前に設定されていることを確認してください。このリンク制限は、XAの動的登録(Oracle XAスイッチxaoswd)を使用する場合に必要です。

Oracle DatabaseのXAコールは、クライアント共有ライブラリ(プラットフォームに応じてlibclntsh.a、libclntsh.sl、libclntsh.soまたはlibclntsh.dylib)およびクライアント静的ライブラリ(libclntst11.a)の両方で定義されています。これらのライブラリは、$ORACLE_HOME/libディレクトリにあります。

7 SQL*LoaderおよびPL/SQLのデモ

この章では、Oracle Databaseとともに使用できるSQL*LoaderとPL/SQLの各デモ・プログラムを作成および実行する方法について説明します。内容は次のとおりです。

- SQL*Loaderのデモ
- PL/SQLのデモ
- 64ビットOracle Database PL/SQLからの32ビット外部プロシージャのコール

> ノート:
>
> この章で説明するデモンストレーションを使用するには、Oracle Database 19c Examples メディアに収録されている Oracle Database Examples をインストールする必要があります。デモンストレーションを作成する前に、JONES アカウントのロックを解除してパスワードを設定する必要があります。

7.1 SQL*Loaderのデモ

SQL*Loaderのデモを実行する場合は、`ulcase.sh`ファイルを実行します。デモを個別に実行する場合は、ファイル内に含まれている情報を読んで、実行方法を確認してください。

7.2 PL/SQLのデモ

PL/SQLには多数のデモ・プログラムが含まれています。これらのプログラムを使用する前に、データベース・オブジェクトを作成し、サンプル・データをロードする必要があります。オブジェクトを作成してサンプル・データをロードするには:

1. ディレクトリをPL/SQLデモ・ディレクトリに変更します。

   ```
   $ cd $ORACLE_HOME/plsql/demo
   ```

2. SQL*Plusを起動し、次のコマンドを入力します。

   ```
   $ sqlplus
   SQL> CONNECT JONES
   Enter password: password
   ```

3. 次のコマンドを実行し、オブジェクトを作成してサンプル・データをロードします。

   ```
   SQL> @exampbld.sql
   SQL> @examplod.sql
   ```

> ノート:
>
> デモは、十分な権限を持つ Oracle ユーザーとして作成してください。デモは、作成時と同じ Oracle ユーザーで実行してください。

PL/SQLカーネル・デモ

次のPL/SQLカーネル・デモは、ソフトウェアとともに使用できます。

- examp1.sqlからexamp8.sql
- examp11.sqlからexamp14.sql
- sample1.sqlからsample4.sql
- extproc.sql

PL/SQLカーネル・デモexampn.sqlまたはsamplen.sqlをコンパイルして実行するには:

1. SQL*Plusを起動し、次のコマンドを入力します。

```
$ cd $ORACLE_HOME/plsql/demo
$ sqlplus
SQL> CONNECT JONES
Enter password: password
```

2. 次のようなコマンドを実行してデモを実行します。demo_nameはデモ名です。

```
SQL> @demo_name
```

extproc.sqlデモを実行するには:

1. 必要に応じて、次のように外部プロシージャのエントリをtnsnames.oraファイルに追加します。

```
EXTPROC_CONNECTION_DATA =
   (DESCRIPTION =
       (ADDRESS_LIST =
          (ADDRESS=(PROTOCOL = IPC)( KEY = EXTPROC))
       )
       (CONNECT_DATA =
          (SID = PLSExtProc)
       )
    )
```

2. 必要に応じて、次のように外部プロシージャのエントリをlistener.oraファイルに追加します。

> ノート:
>
> listener.ora ファイルの SID_NAME に指定する値と、tnsnames.ora ファイルの SID に指定する値は、一致している必要があります。

- Oracle Solaris、LinuxおよびHP-UXの場合:

```
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC=
        (SID_NAME=PLSExtProc)
        (ORACLE_HOME=oracle_home_path)
        (ENVS=EXTPROC_DLLS=oracle_home_path/plsql/demo/extproc.so,
           LD_LIBRARY_PATH=oracle_home_path/plsql/demo)
        (PROGRAM=extproc)
      )
    )
```

- IBM AIX on POWER Systems (64-Bit)の場合:

```
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC=
        (SID_NAME=PLSExtProc)
        (ORACLE_HOME=oracle_home_path)
```

```
        (ENVS=EXTPROC_DLLS=oracle_home_path/plsql/demo/extproc.so,
           LIBPATH=oracle_home_path/plsql/demo)
        (PROGRAM=extproc)
      )
    )
```

3. ディレクトリを$ORACLE_HOME/plsql/demoに変更します。
4. 次のコマンドを実行してextproc.so共有ライブラリを作成し、必要なデータベース・オブジェクトを構築してサンプル・データをロードします。

```
$ make -f demo_plsql.mk extproc.so exampbld examplod
```

データベース・オブジェクトが構築済でサンプル・データがロードされている場合は、次のコマンドを実行します。

```
$ make -f demo_plsql.mk extproc.so
```

5. SQL*Plusから、次のコマンドを実行します。

```
SQL> CONNECT SYSTEM
Enter password: system_password
SQL> GRANT CREATE LIBRARY TO JONES;
SQL> CONNECT JONES
Enter password: password
SQL> CREATE OR REPLACE LIBRARY demolib IS
  2  'oracle_home_path/plsql/demo/extproc.so';
  3  /
```

ノート:

CREATE LIBRARY は非常に高い権限です。この権限は、信頼できるユーザーにのみ付与する必要があります。

6. デモを起動するには、次のコマンドを実行します。

```
SQL> @extproc
```

例7-1 PL/SQLプリコンパイラ・デモ

ノート:

この項で示す make コマンドは、必要なデータベース・オブジェクトを作成し、サンプル・データを JONES スキーマにロードします。

次のプリコンパイラ・デモを利用できます。

- examp9.pc
- examp10.pc
- sample5.pc
- sample6.pc

PL/SQLプリコンパイラ・デモを作成するには、$ORACLE_HOME/libディレクトリを含むようにライブラリ・パス環境変数を設定し、次のコマンドを実行します。

```
$ cd $ORACLE_HOME/plsql/demo
```

```
$ make -f demo_plsql.mk demos
```

デモを1つのみ作成する場合は、makeコマンドにそのデモ名を引数として指定します。たとえば、examp9デモを作成するには、次のコマンドを実行します。

```
$ make -f demo_plsql.mk examp9
```

examp9デモを起動するには、次のコマンドを実行します。

```
$ ./examp9
```

7.3 64ビットOracle Database PL/SQLからの32ビット外部プロシージャのコール

ノート:

この項の内容は、64 ビット Oracle Database にのみ適用されます。

Oracle Database 11gリリース2 (11.2)から、extproc32は64ビットOracle Databaseインストールからは使用できません。したがって、32ビット外部プロシージャを64ビットOracle Databaseから実行する必要がある場合、使用するプラットフォームに対応する32ビット・クライアント・ソフトウェアをインストールして、32ビットextprocを取得する必要があります。特に、カスタム・インストールを32ビット・クライアント・インストールで選択してから、Oracle Databaseユーティリティおよびリスナーを選択する必要があります。

つまり、32ビットextprocを実行するためには、別のOracleホーム(32ビット)が必要です。実行可能ファイル名はextproc32ではなく、単にextprocです。

32ビット外部プロシージャを64ビットOracle Database環境で使用できるようにするには、32ビット・リスナーをextprocに構成し、Oracleホームを(32ビット・クライアント・インストールから)extproc listener.oraエントリに指定する必要があります。

8 Oracle Databaseのチューニング

この章では、Oracle Databaseのチューニング方法について説明します。内容は次のとおりです。

- チューニングの重要性
- オペレーティング・システムのツール
- メモリー管理のチューニング
- ディスク入出力のチューニング
- ディスク・パフォーマンスの監視
- システム・グローバル領域
- オペレーティング・システムのバッファ・キャッシュのチューニング

8.1 チューニングの重要性

この項では、Oracle Databaseを効率的にチューニングし、パフォーマンスを最適化することを目的としています。チューニングを頻繁に行うことで、システム・パフォーマンスが強化され、データのボトルネックの発生を防ぐことができます。

データベースのチューニングを始める前に、「オペレーティング・システムのツール」で説明するツールを使用して、通常の動作を監視する必要があります。

8.2 オペレーティング・システムのツール

データベースのパフォーマンスを評価し、データベース要件を決定できるオペレーティング・システム・ツールがいくつかあります。これらのツールは、Oracleプロセスの統計に加えて、システム全体のCPU使用率、割込み、スワッピング、ページング、コンテキストのスイッチング、I/Oについての統計情報も提供します。

この項では、次に示す共通のツールについて説明します。

- vmstat
- sar
- iostat
- swap_ swapinfo_ swapon_またはlsps
- Oracle Solarisのツール
- Linuxのツール
- IBM AIX on POWER Systems (64-Bit)のツール
- HP-UXのツール

関連項目:

これらのツールの詳細は、オペレーティング・システムのドキュメントおよびmanページを参照してください。

8.2.1 vmstat

vmstatはプロセス、メモリー、ページング、ブロックI/O、トラップ、およびCPUアクティビティに関する情報を報告します。

プロセス、仮想メモリー、ディスク、トラップおよびCPUアクティビティを表示するときは、vmstatコマンドを使用します。表示内容はコマンドで切り替えます。CPUアクティビティのサマリーを5秒間隔で6回表示する場合は、次のいずれかのコマンドを実行します。

- Oracle SolarisおよびHP-UXの場合:

```
$ vmstat -S 5 6
```

ノート:

-S オプションでは、スワッピング統計が表示されます。

- LinuxおよびIBM AIX on POWER Systems (64-bit)の場合:

```
$ vmstat 5 6
```

次に、このコマンドをLinuxで実行した場合の出力例を示します。

```
procs           memory                  swap      io      system       cpu
r b     swpd   free   buff  cache      si so    bi bo   in  cs us   sy id wa st
0 0    130668 103604 198144 5029000     0 0     1 68     8   6  0   0 100 0  0
0 0    130668 103604 198144 5029000     0 0     0 86    226 352 0   0 100 0  0
0 0    130668 103604 198148 5029000     0 0     0 58    223 357 0   0 100 0  0
0 0    130668 103604 198152 5029004     0 0     0 68    223 358 0   0 100 0  0
0 0    130668 103604 198152 5029004     0 0     0 56    223 357 0   0 100 0  0
0 0    130668 103604 198152 5029004     0 0     0 57    228 362 0   0 100 0  0
```

次に、$ vmstat -S 1 2コマンドをHP-UXで実行した場合の出力例を示します。

```
procs     memory              page              faults       cpu
r b w   avm    free    si so pi po fr de sr  in    sy    cs us sy id
1 0 0  112085 2189167  0  0  3  0  0  0  1  1033  32186 108  1  0  98
1 0 0  112085 2189074  0  0  4  0  0  0  0  1022   508   60  0  0 100
```

procs列の下にあるwサブ列は、スワップ・アウトされてディスクに書き込まれたプロセスの数を示します。値が0(ゼロ)以外の場合は、スワッピングが発生してシステムがメモリー不足になっています。

HP-UXおよびOracle Solarisの場合、page列の下にあるsi列とso列は、それぞれ1秒当たりのスワップインおよびスワップアウトのプロセスの数を示します。これらの値はゼロであることが理想的です。

Linuxの場合は、si列とso列がスワップインまたはスワップアウトされたメモリーの量を表します。

page列の下にあるsr列は、スキャン率を示します。利用できるメモリーが不足すると、スキャン率が高くなります。

page列の下にあるpi列およびpo列は、それぞれ1秒当たりのページインとページアウトの回数を示します。ページインおよびページアウトの回数は通常、増加します。使用できるメモリーが十分にあるシステムでも、常に多少のページングは行われます。

関連項目:

出力の解釈については、プラットフォームのvmstat manページを参照してください

8.2.2 sar

オペレーティング・システムのアクティビティ・カウンタの累計を表示するときは、sar(system activity reporter)コマンドを使用します。表示内容はコマンドで切り替えます。

UNIXシステムの場合、次のコマンドは、入出力アクティビティのサマリーを10秒ごとに表示します。

```
$ sar -b 10 10
```

次に、このコマンドのLinuxシステムでの出力例を示します。

```
10:28:01       tps      rtps      wtps   bread/s   bwrtn/s
10:28:11     17.20      0.00     17.20      0.00    300.80
10:28:21     46.40      0.00     46.40      0.00    467.20
10:28:31     16.40      0.00     16.40      0.00    283.20
10:28:41     15.60      0.00     15.60      0.00    275.20
10:28:51     17.02      0.00     17.02      0.00    254.65
10:29:01     35.80      0.00     35.80      0.00    414.40
10:29:11     15.80      0.00     15.80      0.00    273.60
10:29:21     17.40      0.00     17.40      0.00    262.40
10:29:31     32.20      0.00     32.20      0.00    406.40
10:29:41     20.98      0.00     20.98      0.00    354.85
Average:        23.48      0.00     23.48      0.00    329.28
```

sar出力は、ある時点におけるシステムの入出力アクティビティのスナップショットを提供します。複数のオプションを使用して時間隔を指定すると、出力の読取りができなくなることがあります。時間隔を4以下に指定すると、sarアクティビティ自体が出力に影響を与えることがあります。

関連項目:

sarの詳細は、manページを参照してください。

8.2.3 iostat

端末およびディスクのアクティビティを表示するときは、iostatコマンドを使用します。表示内容は、コマンドで切り替えます。iostatコマンドの出力には、ディスク要求キューは表示されず、ビジー状態のディスクが表示されます。この情報は、入出力負荷のバランスを調整する場合に役立ちます。

次のコマンドは、端末およびディスク・アクティビティを5秒間隔で5回表示します。

```
$ iostat 5 5
```

次に、このコマンドをOracle Solarisで実行した場合の出力例を示します。

```
tty          blkdev0           sd1           sd2           sd3           cpu
 tin tout kps tps serv  kps tps serv  kps tps serv  kps tps serv  us sy st id
   0    1   0   0    0    0   0   31    0   0   18    3   0   42   0  0  0 99
   0   16   0   0    0    0   0    0    0   0    0    1   0   14   0  0  0 100
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
   0   16   0   0    0    0   0    0    0   0    0    0   0    0   0  0  0 100
   0   16   0   0    0    0   0    0    2   0   14   12   2   47   0  0  1 98
```

大きなディスク要求キューを調べるときは、iostatコマンドを使用します。要求キューは、特定のディスク・デバイスに対する入出力要求が実行されるまでにかかる時間を示します。要求キューが発生する原因は、特定のディスクに対する入出力要求のボリュームが大きいこと、または入出力の平均シーク時間が長いことです。ディスク要求キューは、0(ゼロ)またはそれに近い値であることが理想的です。

8.2.4 swap、swapinfo、swaponまたはlsps

スワップ領域の使用量に関する情報を表示するときは、swap、swapinfo、swaponまたはlspsの各コマンドを使用します。スワップ領域が不足するとプロセスの応答が停止し、Out of Memoryエラーでプロセスが生成できなくなることがあります。次の表は、プラットフォームごとの適切なコマンドの一覧です。

プラットフォーム	コマンド
Oracle Solaris	swap -lh、swap -sh および zfs list rpool/swap
Linux	swapon -s
POWER Systems (64 ビット) 上の IBM AIX	lsps -a
HP-UX	swapinfo -m

次に、swap -l commandコマンドをOracle Solarisで実行した場合の出力例を示します。

```
swapfile                      dev     swaplo    blocks     free
/dev/zvol/dsk/rpool/swap    274,1       8       20971447    20971447
```

8.2.5 Oracle Solarisのツール

Oracle Solarisシステムでは、mpstatコマンドを使用して、マルチプロセッサ・システムの各プロセッサの統計を表示します。表の各行は、1プロセッサのアクティビティを示します。1行目は、システムが再起動してからのすべてのアクティビティをまとめて表示します。後続の各行は、前の時間隔のアクティビティをまとめて表示します。すべての値は、特に明記されていない場合は、1秒当たりのイベント数です。引数は、統計および反復回数の間の時間隔です。

次に、mpstatコマンドを実行した場合の出力例を示します。

```
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  st idl
  0   3   0    2   319  103  109    0   19   11    0   73     0   1   0  99
  1   4   0    1    71    1   95    0   14   12    0   96     0   0   0  99
```

8.2.6 Linuxのツール

Linuxシステムでは、top、freeおよびcat /proc/meminfoコマンドを使用して、スワップ領域、メモリーおよびバッファの使用量を表示します。

8.2.7 IBM AIX on POWER Systems (64-Bit)のツール

次の項では、IBM AIX on POWER Systems (64-bit)上で使用可能なツールについて説明します。

- Base Operation Systemツール
- Performance Toolbox
- System Management Interface Tool

関連項目:

これらのツールの詳細は、IBM AIX on POWER Systems (64-bit)オペレーティング・システムのドキュメントおよびmanページを参照してください。

8.2.7.1 Base Operation Systemツール

IBM AIX on POWER Systems (64-bit)のBase Operation Systemには、UNIXシステムに以前から含まれていたパフォーマンス・ツールや、IBM AIX on POWER Systems (64-bit)の実装固有の機能を管理するために必要なパフォーマンス・ツールが含まれます。次の表に、最も重要なBase Operation Systemツールを示します。

ツール	機能
lsattr	デバイスの属性を表示します。
lslv	論理ボリューム、つまり物理ボリュームに対する論理ボリュームの割当てに関する情報を表示します。
netstat	ネットワーク関連のデータ構造の内容を表示します。
nfsstat	ネットワーク・ファイル・システムおよびリモート・プロシージャ・コールのアクティビティに関する統計を表示します。
nice	プロセスの初期優先順位を変更します。
no	ネットワーク・オプションを表示または設定します。
ps	1 つ以上のプロセスのステータスを表示します。
reorgvg	ボリューム・グループ内の物理パーティション割当てを再編成します。
time	経過した実行時間、ユーザーの CPU 処理時間およびシステムの CPU 処理時間を表示します。
trace	選択したシステム・イベントを記録および報告します。
vmo	仮想メモリー・マネージャのチューニング可能なパラメータを管理します。

8.2.7.2 Performance Toolbox

IBM AIX on POWER Systems (64-bit)のPerformance Toolboxには、システム・アクティビティをローカルおよびリモートで監視しチューニングするためのツールが含まれます。Performance Tool Boxは、Performance Tool Box ManagerおよびPerformance Tool Box Agentの2つのコンポーネントで主に構成されています。Performance Tool Box Managerは、xmperfユーティリティを使用して、構成内の様々なシステムからデータを収集および表示します。Performance Tool Box Agentは、xmserdデーモンを使用して、Performance Tool Box Managerからデータを収集しデータを転送します。Performance Tool Box Agentは、Performance Aide for IBM AIX on POWER

Systems (64-bit)と呼ばれる個別の製品としても使用できます。

Performance Tool BoxおよびPerformance Aideの両方に、次の表に示す監視およびチューニング・ツールが含まれます。

ツール	説明
fdpr	特定のワークロードに対する実行可能プログラムを最適化します。
filemon	トレース機能を使用して、ファイル・システムのアクティビティを監視および報告します。
fileplace	論理または物理ボリューム内のファイルのブロックの配置を表示します。
lockstat	カーネル・ロックの競合に関する統計を表示します。
lvedit	ボリューム・グループ内の論理ボリュームを対話方式で配置します。
netpmon	トレース機能を使用して、ネットワークの入出力およびネットワーク関連の CPU 使用率を報告します。
rmss	システムを様々なメモリー・サイズでシミュレートし、パフォーマンスをテストします。
svmon	仮想メモリーの使用量に関する情報を取得および分析します。
syscalls	システム・コールを記録し、件数をカウントします。
tprof	trace 機能を使用して、モジュールおよびソース・コード文レベルの CPU 使用率を報告します。
BigFoot	プロセスのメモリー・アクセスのパターンを報告します。
stem	サブルーチン・レベルのエントリを許可し、既存の実行可能ファイルのインスツルメント処理を終了します。

関連項目:

- これらのツールの詳細は、『Performance Toolbox Version 2 and 3 Guide and Reference』を参照してください。
- これらのツールの構文については、『AIX 5L Performance Management Guide』を参照してください。

8.2.7.3 System Management Interface Tool

IBM AIX on POWER Systems (64-bit)のSystem Management Interface Tool(SMIT)は、様々なシステム管理およびパフォーマンス・ツールに、メニュー方式のインタフェースを提供します。SMITを使用すると、様々なツールにナビゲートし、実行するジョブに集中できます。

8.2.8 HP-UXのツール

次のパフォーマンス分析ツールが、HP-UXシステムで使用できます。

- GlancePlus/UX

  このHP-UXユーティリティは、システムのアクティビティを測定するオンライン診断ツールです。GlancePlusでは、システム・リソースの使用状況に関する情報が表示されます。システムの入出力、CPUおよびメモリー使用量に関する動的な情報が、一連の画面に表示されます。このユーティリティを使用して、プロセスごとのリソースの使用状況を監視できます。

- HP Programmer's Analysis Kit

  HP Programmer's Analysis Kitには、次のツールが含まれます。

  - Puma

    このツールは、プログラムの実行中にパフォーマンス統計を収集します。グラフ表示により、収集した統計を表示および分析できます。

  - Thread Trace Visualizer

    このツールは、インスツルメント処理スレッド・ライブラリ、`libpthread_tr.sl`により作成されるトレース・ファイルを、グラフ表示します。これにより、スレッドの相互作用を表示し、スレッドがブロックされてリソースを待機している箇所を確認できます。

    HP Programmer's Analysis Kitは、HP Fortran 77、HP Fortran 90、HP C、HP C++、HP ANSI C++およびHP Pascalのコンパイラに同梱されています。

    次の表に、HP-UXでパフォーマンス・チューニングの追加に使用できるパフォーマンス・チューニング・ツールを示します。

ツール	説明
`caliper`(Itanium のみ)	高速の動的インスツルメント処理とともに、キャッシュ・ミス、Translation Look-aside Buffer または命令サイクルなどの分析タスクのためにランタイム・アプリケーション・データを収集します。これは、C、C++、Fortran およびアセンブリ・アプリケーション用の動的パフォーマンス測定ツールです。
`gprof`	プログラムの実行プロファイルを作成します。
`monitor`	プログラム・カウンタを監視し、特定の関数をコールします。
`netfmt`	ネットワークを監視します
`netstat`	ネットワーク・パフォーマンスに関する統計を報告します。
`nfsstat`	ネットワーク・ファイル・システムおよびリモート・プロシージャ・コールのアクティビティに関する統計を表示します。

ツール	説明
nettl	ロギングおよびトレースにより、ネットワーク・イベントまたはパケットを取得します。
prof	C プログラムの実行プロファイルを作成してプログラムのパフォーマンス統計を表示し、プログラムの実行時間の大半が消費されている箇所を示します。
profil	プログラム・カウンタ情報をバッファにコピーします。
top	システム上の上位プロセスを表示し、情報を定期的に更新します。

8.3 メモリー管理のチューニング

メモリー・チューニング・プロセスでは、最初にページングおよびスワッピング領域を測定して、使用可能なメモリー量を確認します。システムのメモリー使用量の確認後、Oracleバッファ・キャッシュをチューニングします。

Oracleバッファ・マネージャによって、アクセス頻度の最も高いデータをキャッシュに長く保存できます。バッファ・マネージャを監視し、バッファ・キャッシュをチューニングすると、Oracle Databaseのパフォーマンスが大幅に向上することがあります。各システムのOracle Databaseバッファ・サイズの最適値は、システム全体の負荷や他のアプリケーションと比較した場合のOracle Databaseの優先順位によって異なります。

この項には次のトピックが含まれます:

- 十分なスワップ領域の割当て
- 監視ページング
- Oracleブロック・サイズの調整
- メモリー・リソースの割当て

8.3.1 十分なスワップ領域の割当て

スワッピングは、オペレーティング・システムのオーバーヘッドに大きく影響するため、最小限に抑える必要があります。スワッピングが行われているかどうかを調べるには、sarコマンドまたはvmstatコマンドを使用します。これらのコマンドで使用するオプションについては、manページを参照してください。

システムでスワッピングが行われている場合は、メモリーを節約するために、次の処理を行います。

- 必要以上にシステム・デーモン・プロセスまたはアプリケーション・プロセスを実行しないようにします。
- データベース・バッファの数を減らし、一部のメモリーを解放します。
- オペレーティング・システム・ファイル・バッファの数を減らします。

スワップ領域の使用量を確認するには、プラットフォームに応じて、次のいずれかのコマンドを実行します。

プラットフォーム	コマンド
Oracle Solaris	`swap -l`、`swap -s` および `zfs get volsize rpool/swap`

プラットフォーム	コマンド
Linux	`swapon -s`
POWER Systems (64 ビット) 上の IBM AIX	`lsps -a`
HP-UX	`swapinfo -m`

スワップ領域の使用量を監視し、必要に応じて値を大きくしてください。次の表では、インストールされているRAMと構成済スワップ領域要件の、初期に推奨される関係について説明します。

RAM	スワップ領域
1 から 2GB	RAM のサイズの 1.5 倍
2GB から 16GB	RAM のサイズと同じ
16GB 超	16GB

スワップ領域をシステムに追加するには、プラットフォームに応じて、次のいずれかのコマンドを実行します。

プラットフォーム	コマンド

Oracle Solaris	次のオプションのいずれかを使用します。
	● ZFS ファイル・システムの場合は、次のコマンドを使用してスワップ・ボリュームを増やします。
	`zfs set volsize=newsize rpool/swap`
	`newsize` は、増やすスワップ・ボリュームのサイズです。
	● ZFS 以外のファイル・システムの場合は、次のコマンドを使用します。
	`swap -a`

Linux	`swapon -a`
POWER Systems (64 ビット) 上の IBM AIX	`chps` または `mkps`
HP-UX	`swapon`

関連項目:

- My Oracle Supportノート1587357.1
- これらのコマンドの詳細は、オペレーティング・システムのドキュメントを参照してください。

ノート:

12c 以降の Oracle Database では、Oracle Solaris の Optimized Shared Memory (OSM)モデルを使用して、自動メモリー管理を実装します。DISM とは異なり、OSM ではディスクのスワップ領域を二重に割り当てる必要はありません。スワップ領域の要件については、次のノートを参照してください:

My Oracle Support ノート 1010818.1

8.3.2 監視ページング

プログラムを実行するためにプログラム全体をメモリーに格納しておく必要はないため、ページングはスワッピングほど深刻な問題ではありません。少量のページアウトでは、システムのパフォーマンスにほとんど影響はありません。

大量のページングを検出するには、高速応答時またはアイドル時の測定値と、低速応答時の測定値を比較します。

ページングを監視するには、`vmstat`コマンドおよび`sar`コマンドを使用します。

関連項目:

プラットフォームの監視結果の解釈については、manページまたはオペレーティング・システムのドキュメントを参照してください。

Oracle Solarisでは、`vmstat –p`はアドレス変換ページ・フォルトの数を示します。アドレス変換フォルトは、メモリー内にない有効なページをプロセスが参照した場合に発生します。

メモリーに関連する問題を分析するには、`vmstat`出力の空きメモリーの量を調べることから始める必要があります。空きメモリーが少ない場合は、`sr` (scan rate)列がゼロ以外の値になっていないか確認する必要があります。これは、ページ・スキャナがメモリー・ページをスキャンして、空きリストに戻して再利用することを示しています。

匿名(不良)ページングは、`api` (anonymous page-in)列と`apo` (anonymous page-out)列の下の`vmstat -p`列の出力で監視できます。この種類のページングは、メモリー不足の間にシステムが匿名ページをスワップ・デバイスに移動すると発生します。

システムで大量のページアウト・アクティビティが常に発生している場合は、次の方法で解決してください。

- メモリーを増設します。
- 一部の作業を別のシステムに移します。
- システム・グローバル領域(SGA)で使用するメモリーを少なく設定します。

関連項目

- MOSドキュメント: 1007494.1

8.3.3 Oracleブロック・サイズの調整

読取り操作時には、オペレーティング・システムのブロック全体がディスクから読み取られます。データベースのブロック・サイズが、オ

ペレーティング・システムのファイル・システムのブロック・サイズより小さい場合は、入出力バンド幅の効率が悪くなります。Oracle Databaseのブロック・サイズをファイル・システムのブロック・サイズの倍数になるように設定すると、パフォーマンスを最大5%向上させることができます。

データベースのブロック・サイズは、DB_BLOCK_SIZE初期化パラメータで設定します。ただし、このパラメータの値を変更するには、データベースを再作成する必要があります。

DB_BLOCK_SIZEパラメータの現在の設定値を調べるには、SQL*PlusのSHOW PARAMETER DB_BLOCK_SIZEコマンドを実行します。

8.3.4 メモリー・リソースの割当て

パラメータを設定して、ワークロードの要件、および同じシステムで稼働している様々なデータベース・インスタンスの要件に基づいて、自動的にメモリーを割り当てることができます。MEMORY_TARGETパラメータは、そのインスタンスに対してOracleシステム全体の使用可能メモリーを指定し、SGAおよびProcess Global Area(PGA)コンポーネントを自動的にチューニングします。MEMORY_MAX_TARGETパラメータは、MEMORY_TARGETパラメータが動的に増加できる上限値を示します。

デフォルトでは、これらのパラメータの値は両方とも0であり、自動チューニングは行われません。自動チューニングをアクティブにするには、MEMORY_TARGETパラメータを0以外の値に設定します。MEMORY_TARGETパラメータを動的に有効にするには、起動時にMEMORY_MAX_TARGETパラメータを設定する必要があります。

ノート:

MEMORY_TARGET パラメータを 0 以外の値に設定すれば、MEMORY_MAX_TARGET パラメータは自動的にこの値を取得します。

MEMORY_TARGET パラメータおよび MEMORY_MAX_TARGET パラメータは、Linux、Oracle Solaris、HP-UX および IBM AIX on POWER Systems (64-bit)のプラットフォームでのみサポートされています。

Oracle Solaris では、MEMORY_TARGET または MEMORY_MAX_TARGET で動的緊密共有メモリーを使用できます。詳細は、「Oracle Solaris での Oracle Database の管理」を参照してください。

Linux では、MEMORY_TARGET または MEMORY_MAX_TARGET が有効になっている場合、一部の共有リソース要件が増加します。詳細は、「共有リソースの割当て」の項を参照してください。

ヒント:

MEMORY_TARGET および MEMORY_MAX_TARGET パラメータは、元の設定、コンピュータ上の Oracle に使用可能なメモリー、およびワークロードのメモリー要件に基づいて設定できます。

8.4 ディスク入出力のチューニング

使用可能なディスク全体で入出力を均等に分散して、ディスクへのアクセス時間が短くなるようにしてください。小規模なデータベースやRAIDを使用しないデータベースでは、それぞれのデータファイルと表領域を使用可能なディスク間に分散してください。

この項では、次の項目について説明します。

- 自動ストレージ管理の使用
- 適切なファイル・システム・タイプの選択

8.4.1 自動ストレージ管理の使用

データベース記憶域に自動ストレージ管理を使用すると、すべてのデータベース入出力が、自動ストレージ管理ディスク・グループ内の使用可能なすべてのディスク・デバイス間に分散されます。

自動ストレージ管理を使用することで、ディスク入出力を手動でチューニングする必要がなくなります。

8.4.2 適切なファイル・システム・タイプの選択

オペレーティング・システムに応じて、いくつかのファイル・システム・タイプから選択できます。ファイル・システム・タイプごとにそれぞれ特性が異なります。このことが、データベースのパフォーマンスに大きな影響を与えます。次の表に、一般的なファイル・システム・タイプを示します。

ファイル・システム	プラットフォーム	説明
ZFS	Oracle Solaris	Oracle Solaris ZFS ファイル・システム
S5	HP-UX	UNIX System V ファイル・システム
UFS	Oracle Solaris、IBM AIX on POWER Systems (64-Bit)および HP-UX	Unix ファイル・システム(BSD UNIX から派生)
VxFS	Oracle Solaris、IBM AIX on POWER Systems (64-Bit)および HP-UX	VERITAS ファイル・システム
ext2/ext3	Linux	Linux 用拡張ファイル・システム
OCFS2	Linux	Oracle Cluster ファイル・システム
JFS/JFS2	POWER Systems (64 ビット)上の IBM AIX	ジャーナル・ファイル・システム
GPFS	POWER Systems (64 ビット)上の IBM AIX	一般的なパラレル・ファイル・システム

ファイル・システムとアプリケーションには、必ずしも互換性があるとはかぎりません。たとえば、UFSの異なる実装間でも、互換性の比較は難しくなります。選択したファイル・システムによって、パフォーマンスに最大20%の開きが出る場合があります。ファイル・システムを使用する場合は、次のことを行ってください。

- ハードディスクがクリーンで断片化されないように、ファイル・システムのパーティションを新しく作成します。

- データベース・ファイルに対してファイル・システムを使用する前に、パーティションでファイル・システム・チェックを行います。
- ディスク入出力をできるだけ均等に分散します。
- 論理ボリューム・マネージャまたはRAIDデバイスを使用していない場合、データファイルとは異なるファイル・システムにログ・ファイルを格納することを検討してください。

関連項目:

ZFSのOracle Database向けチューニングの詳細は、Oracle Solaris 11.3チューニング可能パラメータのリファレンス・マニュアルの「データベース製品に向けたZFSのチューニング」を参照してください。 .

8.5 ディスク・パフォーマンスの監視

次の項では、ディスク・パフォーマンスの監視方法について説明します。

- その他のオペレーティング・システムでのディスク・パフォーマンスの監視
- ディスク再同期化の使用による自動ストレージ管理ディスク・グループの監視

8.5.1 オペレーティング・システムでのディスク・パフォーマンスの監視

ディスク・パフォーマンスを監視するには、`sar -b`および`sar -u`コマンドを使用します。

次の表に、`sar -b`コマンド出力の列をいくつか示します。これらの列は、ディスク・パフォーマンスの分析に重要です。

列	説明
`bread/s`、`bwrit/s`	1 秒ごとに読み取られるブロック数と書き込まれるブロック数(ファイル・システム・データベースに重要)
`pread/s`、`pwrit/s`	RAW デバイス上の 1 秒ごとの I/O 操作の数(RAW パーティション・データベース・システムに重要)

キー・インジケータは次のとおりです。

- `bread`、`bwrit`、`pread`および`pwrit`列の値の合計は、ディスク入出力サブシステムのアクティビティのレベルを示します。合計値が大きいほど、入出力サブシステムはビジーになります。物理ドライブの数が多いほど、合計のしきい値が高くなる可能性があります。
- `%rcache`列の値は91以上、`%wcache`列の値は61以上である必要がありますそれ以外の場合は、システムがディスク入出力バウンドになる可能性があります。

8.5.2 ディスク再同期化の使用による自動ストレージ管理ディスク・グループの監視

`alter diskgroup disk online`および`alter diskgroup disk offline`コマンドを使用して、ディスク・セットへの入出力を一時的に停止します。これらのコマンドを使用すると、定期的なメンテナンス・タスクを実行したり、ディスク・ファームウェアのアップグレードなどのアップグレードを実行することができます。ディスク・グループ内の一部のディスクで一時的な障害が発生した場合は、`alter diskgroup disk online`を使用して、ディスク・グループを迅速にリカバリします。

8.6 システム・グローバル領域

SGAとは、共有メモリーに格納されているOracle構造体のことです。これには、静的データ構造体、ロックおよびデータ・バッファが含まれています。

単一の共有メモリー・セグメントの最大サイズはshmmaxカーネル・パラメータで指定されます。

次の表に、このパラメータの推奨値をプラットフォームごとに示します。

プラットフォーム	推奨値
Oracle Solaris	4294967295 または 4GB - 16MB。
Linux	次の値のうち小さいほう。 ● システムに搭載された物理メモリーの半分のサイズ。 ● 4GB - 1 バイト
POWER Systems (64 ビット)上の IBM AIX	NA
HP-UX	システムに搭載された物理メモリーのサイズ。

SGAのサイズが共有メモリー・セグメントの最大サイズ(shmmaxまたはshm_max)を超える場合、Oracle Databaseでは、要求されたSGAサイズになるように、連続したセグメントが連結されます。shmsegカーネル・パラメータには、任意のプロセスで連結できるセグメントの最大数を指定します。SGAのサイズを制御するには、次の初期化パラメータを設定します。

- DB_CACHE_SIZE
- DB_BLOCK_SIZE
- JAVA_POOL_SIZE
- LARGE_POOL_SIZE
- LOG_BUFFERS
- SHARED_POOL_SIZE

または、SGAサイズを自動的にチューニングできるように、SGA_TARGET初期化パラメータを設定します。

これらのパラメータの値は、十分注意して設定してください。値を大きく設定しすぎると、物理メモリーに対する共有メモリーの割合が非常に高くなります。そのため、パフォーマンスが低下します。

共有サーバーで構成されているOracle Databaseでは、SHARED_POOL_SIZE初期化パラメータの値を大きく設定するか、LARGE_POOL_SIZE初期化パラメータを使用したカスタム構成が必要です。Oracle Universal Installerを使用してデータベースをインストールした場合、SHARED_POOL_SIZEパラメータの値は、Oracle Database Configuration Assistantによって自動的に設定されます。ただし、データベースを手動で作成した場合は、パラメータ・ファイルでSHARED_POOL_SIZEパラメータの値を同時ユーザーごとに1KBずつ増やしてください。

各Oracleプロセスが SGA全体をアドレス指定するためには、十分な共有メモリーが必要です。

- SGAサイズの確認

- システム・リソース検証ユーティリティ
- セマフォ・パラメータの設定に関するガイドライン
- IBM AIX on POWER Systems (64-Bit)での共有メモリー

8.6.1 SGAサイズの確認

次のいずれかの方法で、SGAサイズを確認できます。

- 次のSQL*Plusコマンドを実行して、実行中のデータベースのSGAサイズを表示します。

  ```
  SQL> SHOW SGA
  ```

  結果はバイト単位で表示されます。

- データベース・インスタンスを起動して、システム・グローバル領域の合計ヘッダーの横にSGAのサイズを表示します。
- oracleユーザーでipcsコマンドを実行します。

8.6.2 システム・リソース検証ユーティリティ

システム・リソース検証ユーティリティ(sysresv)は、Oracle8i以上のリリースで使用できます。これを使用すると、指定したOracleシステム識別子(ORACLE_SID)に対してOracleインスタンスおよびオペレーティング・システム・リソースの情報が提供されます。このユーティリティは$ORACLE_HOME/binにありますが、他の場所から使用できます。

8.6.2.1 sysresvユーティリティの目的

sysresvユーティリティを使用して、Oracleインスタンスのステータスを表示したり、そのインスタンスが使用するオペレーティング・システム・リソース(メモリーやセマフォ・パラメータなど)を特定します。このユーティリティは、複数のインスタンスが実行されている場合に特に役立ちます。たとえば、インスタンスが応答しない場合は、このユーティリティを使用してオペレーティング・システム・リソースを削除できます。

Oracleインスタンスがクラッシュしたか、強制終了された場合にこのインスタンスに関連するメモリーおよびセマフォが自動的にクリーンアップされなかったときに、このユーティリティを使用できます。このユーティリティは、実行されているOracleインスタンスを判断する際にも役立ちます。

8.6.2.2 sysresvを使用するための条件

sysresvユーティリティを使用するには、システム・グローバル領域(SGA)へのアクセス権が必要です。SGAにアクセスするには、Oracle所有者またはOracleバイナリを所有するグループのメンバーである必要があります。

8.6.2.3 sysresvの構文

sysresvユーティリティの構文は次のとおりです。

```
sysresv [-i] [-f] [-d on|off] [-l sid1[ sid2 ...]]
```

説明:

- -i: 各sidのIPCリソースを削除する前に確認を求めます。
- -f: 確認を求めずにIPCリソースを削除します。このフラグは-iオプションよりも優先されます
- -d on|off : onの場合は各sidのIPCリソースを一覧表示します。指定しない場合、-dのデフォルトはonです
- -l sid1 [sid2 sid3]: 空白区切りの1つ以上のシステム識別子に対してsysresvチェックを実行します

sysresvをフラグなしで使用した場合、環境変数のOracleインストール所有者ユーザー・プロファイル・リストの$ORACLE_SID環境変数によって識別されるOracleインスタンスのIPCリソースがレポートされます。

8.6.2.4 sysresvの使用例

次の例では、sysresvユーティリティを使用する方法を示します。

```
$ sysresv
IPV Resources for ORACLE_SID "sales" :
Shared Memory:
ID                          KEY
10345                    0x51c051ad
Semaphores
ID
10345                    0x51c051ad
Oracle Instance alive for sid "sales"
```

8.6.3 セマフォ・パラメータの設定に関するガイドライン

デフォルトのセマフォ・パラメータ値が小さすぎてすべてのOracleプロセスに対応できない場合にのみ、次のガイドラインを使用してください。

ノート:

セマフォ・パラメータの設定方法の詳細は、オペレーティング・システムのドキュメントを参照することをお薦めします。

1. 次の計算式を使用して、全体的な最小セマフォ要件を計算します。

```
sum (process parameters of all database instances on the system) + overhead for
oracle background processes + system and other application requirements
```

2. semmns (サーバーで許容できる最大セマフォ数)をこの合計に設定します。
3. semmsl (1つのセマフォ・セットの最大セマフォ数)を250に設定します。
4. semmni (最大セマフォ・セット数)を、semmns/semmslを切り上げて最も近い1024の倍数にした値に設定します。

ノート:

Oracle Solaris はこれらのセマフォ・パラメータを使用せず、リソース・コントロールを使用します。My Oracle Support ノート 1006158.1 を参照してください。

```
https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1
006158.1
```

関連項目:

次のURLにあるMy Oracle Supportノート226209.1「Linux: How to Check Current Shared Memory, Semaphore Values」を参照してください。

https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=226209.1

8.6.4 IBM AIX on POWER Systems (64-Bit)での共有メモリー

ノート:

この項の内容は、IBM AIX on POWER Systems (64-bit)にのみ適用されます。

共有メモリーでは、プロセス間で共通の仮想メモリー・リソースが使用されます。プロセスは、パフォーマンスを向上させるために、表やキャッシュ済エントリなどの仮想メモリー変換リソース・セットを介して仮想メモリー・セグメントを共有します。

共有メモリーを固定して、ページングを防ぎ、入出力のオーバーヘッドを減らすことができます。そのためには、LOCK_SGAパラメータをtrueに設定します。IBM AIX on POWER Systems (64-bit) 5Lの場合、基礎となるハードウェアでラージ・ページ機能がサポートされていれば、同じパラメータによりこの機能がアクティブになります。

次のコマンドを実行し、固定されたメモリーをOracle Databaseで使用できるようにします。

```
$ /usr/sbin/vmo -r -o v_pinshm=1
```

設定する実メモリーの最大の割合がpercent_of_real_memoryの場合、次のようなコマンドを実行し、固定されたメモリーで使用可能な実メモリーの最大量を設定します。

```
$ /usr/sbin/vmo -r -o maxpin percent=percent_of_real_memory
```

maxpin percentオプションを使用する場合、固定できる空きメモリーをカーネルで使用可能にするためには、固定されたメモリーの量がOracle SGAのサイズを、システム上の実メモリーの3%以上は超えている必要があります。たとえば、2GBの物理メモリーで、400MB(RAMの20%)のSGAを固定する場合、次のコマンドを実行します。

```
$ /usr/sbin/vmo -r -o maxpin percent=23
```

ノート:

デフォルトの maxpin percent 値は 80 percent に設定され、ほとんどのインストールで機能します。

svmonコマンドを実行し、システムの操作時に固定されたメモリーの使用量を監視します。LOCK_SGAパラメータがtrueに設定されている場合にのみ、Oracle Databaseはメモリーの固定を実行します。SGAサイズが、固定に使用できるメモリーのサイズを超える場合、これらのサイズを超えるSGAの部分が、通常の共有メモリーに割り当てられます。

IBM AIX on POWER Systems (64-Bit) POWER4およびPOWER5ベース・システムでのラージ・ページ機能

POWER4またはPOWER5システム上で10件の16MBのラージ・ページをオンにし、予約するには、次のコマンドを実行します。

```
$ /usr/sbin/vmo -r -o lgpg_regions=10 -o lgpg_size=16777216
```

このコマンドを実行すると、bosbootの実行を促すメッセージが表示されます。また、変更を有効にするには再起動が必要という警告が表示されます。

SGA全体を格納できる十分な大きさのラージ・ページを指定することをお薦めします。LOCK_SGAパラメータがtrueに設定されている場合、Oracle Databaseインスタンスはラージ・ページの割当てを実行します。

16MBのページが常に固定され、これは標準メモリーには使用できません。16MBサイズのページのプールが構成されている場合、このメモリーは、他のアプリケーションがラージ・ページを現在使用していなくても標準メモリーの割当てに使用できません。

POWER5ベース・システムでは64Kページがサポートされます。Oracleでは、使用可能な場合は、これらのページをSGAに使用します。これらの64Kページには、追加構成が必要ありません。また、LOCK_SGAパラメータ設定に依存しません。

ラージ・ページの使用を監視するには、次のコマンドを使用します。

```
$ vmstat -P all
```

IBM AIX on POWER Systems (64-bit)オペレーティング・システムで16MBページまたは固定されたメモリー(共有メモリーの割当て時)を使用する場合、OracleユーザーIDには、CAP_BYPASS_RAC_VMMおよびCAP_PROPAGATE機能が必要です。データベース・インスタンスの起動に使用するユーザーIDにも、同じ機能が必要です。特に、Oracle Real Application Cluster(Oracle RAC)データベースでラージ・ページを使用する場合は、Oracle RACデータベース・インスタンスの起動および停止にsrvctlコマンドを使用するため、CAP_BYPASS_RAC_VMMおよびCAP_PROPAGATE機能をrootユーザーにも設定する必要があります。

関連項目:

固定されたメモリーおよびラージ・ページの有効化とチューニングの詳細は、IBM AIX on POWER Systems (64-bit)のドキュメントを参照してください。

これらの機能は、次のコマンドを使用して設定および確認できます。

- 次のコマンドを実行して、現在の機能を確認します。

  ```
  $ lsuser –a capabilities oracle
  ```

- CAP_BYPASS_RAC_VMMおよびCAP_PROPAGATE機能をこのユーザーIDに追加します。

  ```
  $ chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE oracle
  ```

ノート:

機能の属性を表示および設定できるのは、root ユーザーのみです。

8.7 オペレーティング・システムのバッファ・キャッシュのチューニング

Oracle Databaseバッファ・キャッシュのサイズを調整します。メモリーに制限がある場合は、オペレーティング・システムのバッファ・キャッシュも調整します。

オペレーティング・システムのバッファ・キャッシュには、メモリーからディスクまたはディスクからメモリーへの転送中に、メモリー内のデータ・ブロックが保持されます。

Oracle Databaseバッファ・キャッシュは、Oracle Databaseバッファを格納するためのメモリー内の領域です。

システムで使用できるメモリーに制限がある場合は、それに応じてオペレーティング・システムのバッファ・キャッシュのサイズを小さくします。

調整するバッファ・キャッシュを判断するには、sarコマンドを使用します。

A LinuxシステムでのOracle Databaseの管理

この付録では、LinuxシステムでOracle Databaseを管理する方法について説明します。

次の項目が含まれます。

- 非同期入出力のサポート
- 非同期入出力サポート
- 直接入出力サポートの有効化
- 同時マルチスレッドの有効化
- 共有リソースの割当て
- Linuxシステムでのcgroup作成について
- HugePagesの概要

ノート:

Oracle Database 11g リリース 2 (11.2)からは、Linux x86-64 および IBM: Linux on System z メディアには Linux x86 バイナリは含まれません。

A.1 非同期入出力のサポート

ノート:

Linux の自動ストレージ管理では、デフォルトで非同期入出力が使用されます。非同期入出力は、ネットワーク・ファイル・システムに格納されたデータベース・ファイルに対してはサポートされません。

Oracle Databaseでは、カーネルの非同期入出力がサポートされます。デフォルトでは、非同期入出力はRAWボリュームで使用可能です。デフォルトでは自動ストレージ管理で非同期入出力が使用されます。

デフォルトでは、パラメータ・ファイルのDISK_ASYNCH_IO初期化パラメータはTRUEに設定されています。ファイル・システムのファイルに対して非同期入出力を有効にするには:

1. すべてのOracle Databaseファイルが、非同期入出力をサポートしているファイル・システム上にあることを確認します。
2. パラメータ・ファイルのFILESYSTEMIO_OPTIONS初期化パラメータをASYNCHまたはSETALLに設定します。

ノート:

ファイル・システムのファイルが ODM ライブラリ・インタフェースまたは Direct NFS クライアントで管理されている場合、デフォルトで、非同期入出力が有効になります。これらの環境では、非同期入出力を有効にするために、FILESYSTEMIO_OPTIONS を設定する必要はありません。

A.2 非同期入出力サポート

ノート:

Linux の自動ストレージ管理では、デフォルトで非同期入出力が使用されます。非同期入出力は、ネットワーク・ファイル・システムに格納されたデータベース・ファイルに対してはサポートされません。

Oracle Databaseでは、カーネルの非同期入出力がサポートされます。この機能はデフォルトで無効です。

デフォルトでは、RAWデバイス上で非同期I/Oを有効化するため、パラメータ・ファイルのDISK_ASYNCH_IO初期化パラメータはTRUEに設定されています。ファイル・システムのファイルに対して非同期入出力を有効にするには:

1. すべてのOracle Databaseファイルが、非同期入出力をサポートしているファイル・システム上にあることを確認します。
2. パラメータ・ファイル内のFILESYSTEMIO_OPTIONS初期化パラメータをASYNCHに設定すると、非同期入出力が有効になります。非同期入出力と直接入出力の両方を有効にする場合は、パラメータ・ファイル内のFILESYSTEMIO_OPTIONS初期化パラメータをSETALLに設定します。

A.3 直接入出力サポートの有効化

直接入出力サポートは使用可能であり、Linuxでサポートされます。

直接入出力サポートを有効にするには:

- FILESYSTEMIO_OPTIONS初期化パラメータをDIRECTIOに設定します。
- パラメータ・ファイルのFILESYSTEMIO_OPTIONS初期化パラメータをSETALLに設定することにより、非同期入出力および直接入出力の両方を有効にできます。

A.4 マルチスレッド同時処理の有効化

同時マルチスレッドが有効になっている場合、v$osstatビューは、オンライン論理(NUM_LCPUS)および仮想CPU(NUM_VCPUS)に対応した2つの行を追加して報告します。

A.5 共有リソースの割当て

MEMORY_TARGETまたはMEMORY_MAX_TARGET機能を使用するには、次のカーネル・パラメータを変更する必要があります。

- /dev/shmマウント・ポイントは、SGA_MAX_SIZEの値が設定されていれば、その値以上にするか、MEMORY_TARGETとMEMORY_MAX_TARGETのうちいずれか大きいほうの値以上に設定する必要があります。たとえば、MEMORY_MAX_TARGET=4GBのみが設定されている場合、4GBのシステムを/dev/shmマウント・ポイントに作成するには、次の手順を実行します。
  - rootユーザーで次のコマンドを実行します。

    ```
    # mount -t tmpfs shmfs -o size=4g /dev/shm
    ```

  - システムの再起動時にインメモリー・ファイル・システムが確実にマウントされるように、/etc/fstabファイルに次のようなエントリを追加します。

    ```
    tmpfs /dev/shm tmpfs size=4g 0
    ```

- 各Oracleインスタンスのファイル記述子の数は、512*PROCESSESだけ増加します。したがって、ファイル記述子の最大数は、少なくともこの値以上にし、オペレーティング・システムの要件に応じてさらに数を足す必要があります。たとえば、cat /proc/sys/fs/file-maxコマンドで32768が返され、PROCESSESが100である場合、Oracleに対して51200使用可能にするには、rootとしてこの値を6815744以上に設定します。次のどちらかのオプションを使用して、file-max記述子の値を設定します。
  - 次のコマンドを実行します。

    ```
    echo 6815744 > /proc/sys/fs/file-max
    ```

    または
  - /etc/sysctl.confファイル内で次のエントリを変更し、rootとしてシステムを再起動します。

    ```
    fs.file-max = 6815744
    ```
- プロセスごとのファイル記述子の数は、512以上である必要がありますたとえば、rootとして次のコマンドを実行します。
  - bashおよびshの場合

    ```
    # ulimit -n
    ```
  - cshの場合

    ```
    # limit descriptors
    ```

  前のコマンドで200が返される場合、次のコマンドを実行して、プロセッサごとのファイル記述子の上限の値(たとえば、1000)を設定します。
  - bashおよびshの場合

    ```
    # sudo sh
    # ulimit -n 1000
    ```
  - cshの場合

    ```
    # sudo sh
    # limit descriptors 1000
    ```
- LOCK_SGAが有効になっている場合、MEMORY_TARGETおよびMEMORY_MAX_TARGETは使用できません。MEMORY_TARGETおよびMEMORY_MAX_TARGETは、LinuxのHugeページと使用することもできません。

A.6 Linuxシステムでのcgroup作成について

制御グループ(cgroup)は、一連の専用CPUをデータベース・インスタンスに関連付けることによってデータベースのパフォーマンスを向上させます。各データベース・インスタンスはそのcgroupのリソースのみを使用します。

大規模サーバーで統合を行う場合、CPUとメモリーの特定のサブセットにデータベースを制限することをお薦めします。この機能により、Oracle DatabaseインスタンスのCPUとメモリーの制約を簡単に有効にすることができます。

setup_processor_group.shスクリプトを使用してcgroupsを作成します。このスクリプトは、My Oracle Support Webサイトのノート1585184.1からダウンロードします。

https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1585184.1

A.7 HugePagesの概要

HugePagesは、Linuxカーネル2.6に統合された機能です。HugePagesを有効にすると、オペレーティング・システムはデフォルト(通常、4KB)より大きいメモリー・ページをサポートできます。大規模なページ・サイズを使用すると、ページ表エントリへのアクセスに必要なシステム・リソースの量を削減することにより、システム・パフォーマンスを向上させることができます。32ビットおよび64ビット構成の両方で使用できます。HugePagesのサイズは、カーネル・バージョンとハードウェア・アーキテクチャに応じて、2MBから256MBになります。Oracle Databaseの場合、HugePagesを使用すると、ページ状態のオペレーティング・システム・メンテナンスが軽減され、Translation Lookaside Buffer(TLB)ヒット率が向上します。

ノート:

現時点で、透過的な Hugepages は、手動による HugePages の構成の代替手段になりません。

この項には次のトピックが含まれます:

- HugePagesのメモリー割当ての確認
- LinuxでのHugePagesの使用
- HugePagesによるSGAのチューニング
- LinuxでのHugePagesの構成
- HugePages構成の制限
- 透過的なHugePagesの無効化

A.7.1 HugePagesのメモリー割当ての確認

使用するオペレーティング・システムでHugePagesを有効にしている場合は、この情報を確認してください。

Linuxプラットフォームのインストールでは、Oracle Databaseの最高のパフォーマンスを得るためにHugePagesを使用することをお薦めします。HugePagesが有効になっているサーバー上のOracle Grid InfrastructureおよびOracle Databaseをアップグレードする場合、HugePagesのメモリー割当て要件を確認することをお薦めします。

GIMRとHugePagesメモリー

Oracle Grid Infrastructureインストールには、グリッド・インフラストラクチャ管理リポジトリ(GIMR)が含まれます。HugePagesがクラスタ・メンバー・ノードで構成されている場合、GIMRのシステム・グローバル領域(SGA)は、HugePagesメモリーにインストールされます。GIMRのSGAは、HugePagesメモリーを最大1GBまで占有します。Oracle Grid Infrastructureは、クラスタにインストールされたOracle Databaseより前に起動します。

クラスタ・メンバー・ノードのオペレーティング・システムのHugePagesに対するメモリー割当てが、クラスタのすべてのOracle Databaseインスタンスに対応するSGAのサイズにとって不十分な場合、1つ以上のOracle Database SGAがHugePagesではなく通常のページにマップされ、期待するパフォーマンスに到達しないことがあります。この問題を回避するには、アップグレードを計画するときに、HugePages用に予約するメモリーが、メモリー要件に対応するのに十分な大きさであることを確認してください。

クラスタ上でSGAを実行する予定のすべてのデータベースに対して十分な大きさのメモリーをHugePagesに割り当て、HugePagesグリッド・インフラストラクチャ管理リポジトリのSGAに対応します。

A.7.2 LinuxでのHugePagesの使用

Linuxにおいて、Oracle Databaseでラージ・ページ(HugePagesとも呼ばれる)を使用可能にするには、vm.nr_hugepagesカーネル・パラメータの値を設定し、予約するラージ・ページ数を指定します。データベース・インスタンスのSGA全体を保持するために十分なラージ・ページを指定する必要があります。必要なパラメータ値を判断するには、インスタンスのSGAサイズをラージ・ページのサイズで除算してから、結果の端数を切り上げて最も近い整数にします。

デフォルトのラージ・ページ・サイズを判断するには、次のコマンドを実行します。

```
# grep Hugepagesize /proc/meminfo
```

たとえば、/proc/meminfoにラージ・ページのサイズが2MBとリストされ、インスタンスの総SGAサイズが1.6GBの場合は、vm.nr_hugepagesカーネル・パラメータの値を820(1.6GB / 2MB = 819.2)に設定します。

A.7.3 HugePagesによるSGAのチューニング

HugePagesを使用しない場合、オペレーティング・システムではページごとに4KBのメモリーが保持されます。オペレーティング・システムによってページがデータベースのシステム・グローバル領域(SGA)に割り当てられると、オペレーティング・システム・カーネルはSGAに割り当てられた4KBページのそれぞれについて継続的に、ページ表にページ・ライフサイクルを反映する必要があります(ダーティ、空き、プロセスにマッピング済など)。

HugePagesを使用した場合、各ページ表のエントリは2MBから256MBのページを指しているため、オペレーティング・システムのページ表(仮想メモリーから物理メモリーへのマッピング)は小さくなります。

また、カーネルによるライフサイクル監視が必要なページの数が少なくなります。たとえば、64-bitのハードウェアでHugePagesを使用する場合、256MBのメモリーをマップするには、1ページの表エントリ(PTE)が必要です。HugePagesを使用しないで256MBのメモリーをマップするには、256MB×1024KB/4KB=65536PTEが必要です。

HugePagesには次の利点があります。

- TLBヒットが増加することによってパフォーマンスが向上します。
- ページはメモリー内でロックされ、スワップ・アウトが発生しないので、SGAなどの共有メモリー構造のRAMとして使用できます。
- 連続するページを事前に割り当てるため、System V共有メモリー(SGAなど)以外に使用できなくなります。
- ページ・サイズが大きいので、仮想メモリーのうち該当する部分についてカーネルが行う記録作業が少なくなります。

A.7.4 LinuxでのHugePagesの構成

コンピュータにHugePagesを構成するには、次のステップを実行します。

1. 次のコマンドを実行してカーネルでHugePagesがサポートされているかどうかを確認します。

   ```
   $ grep Huge /proc/meminfo
   ```

2. Linuxシステムによっては、HugePagesがデフォルトでサポートされていません。このようなシステムの場合、CONFIG_HUGETLBFSおよびCONFIG_HUGETLB_PAGE構成オプションを使用してLinuxカーネルを構築します。CONFIG_HUGETLBFSは「File Systems」の下にあり、CONFIG_HUGETLB_PAGEはCONFIG_HUGETLBFSを選択すると選択されます。
3. /etc/security/limits.confファイルのmemlock設定を編集します。memlock設定はKB単位で指定し、ロックされるメモリーの上限値を、HugePagesメモリーが有効の場合は現行RAMの90パーセント以上に設定し、

HugePagesメモリーが無効の場合は3145728KB (3GB)以上に設定する必要があります。たとえば、64GBのRAMが搭載されている場合は、次のエントリを追加してメモリー内にロックされるアドレス空間の上限値を大きくします。

```
*    soft    memlock     60397977
*    hard    memlock     60397977
```

memlockの値をSGA要件よりも大きく設定することもできます。

4. oracleユーザーとして再度ログインし、ulimit -lコマンドを実行して新しいmemlock設定を確認します。

```
$ ulimit -l
60397977
```

5. 次のコマンドを実行して、変数Hugepagesizeの値を表示します。

```
$ grep Hugepagesize /proc/meminfo
```

6. 現行の共有メモリー・セグメントのhugepages構成の推奨値を計算するスクリプトを作成するには、次の手順を実行します。

   a. hugepages_settings.shという名前のテキスト・ファイルを作成します。

      hugepages_settings.shスクリプトの作成の詳細は、My Oracle Supportノート401749.1を参照してください。

   b. 次のコマンドを実行して、ファイルの権限を変更します。

      ```
      $ chmod +x hugepages_settings.sh
      ```

7. hugepages_settings.shスクリプトを実行して、hugepages構成の値を計算します。

```
$ ./hugepages_settings.sh
```

ノート:

このスクリプトを実行する前に、hugepages を使用するすべてのアプリケーションが実行中であることを確認します。

8. 次のカーネル・パラメータを設定します。valueは、ステップ7で決定したHugePages値です。

```
# sysctl -w vm.nr_hugepages=value
```

9. HugePagesがシステム再起動後に確実に割り当てられるようにするために、次のエントリを/etc/sysctl.confファイルに追加します。valueは、ステップ7で決定したHugePages値です。

```
vm.nr_hugepages=value
```

10. 次のコマンドを実行して、使用可能なhugepagesを確認します。

```
$ grep Huge /proc/meminfo
```

11. インスタンスを再起動します。

12. 次のコマンドを実行して、使用可能なhugepages(1から2ページの空きページ)を確認します。

```
$ grep Huge /proc/meminfo
```

ノート:

nr_hugepages を使用して HugePages 割当てを設定できない場合は、空きメモリーが断片化している可能性があります。サーバーを再起動すると Hugepages 割当てが有効になります。

A.7.5 HugePages構成の制限

HugePagesには次の制限があります。

- 初期化パラメータMEMORY_TARGETおよびMEMORY_MAX_TARGETの両方の設定を解除する必要があります。たとえば、データベース・インスタンスのパラメータの設定を解除するには、コマンドALTER SYSTEM RESETを使用します。
- 自動メモリー管理(AMM)とHugePagesは、互換性がありません。AMMを使用すると、SGAメモリー全体は/dev/shmの下にファイルを作成することによって割り当てられます。Oracle DatabaseがAMMを使用してSGAを割り当てると、HugePagesは予約されません。Oracle Database 19cでHugePagesを使用するには、AMMを無効にする必要があります。
- VLMを32ビット環境で使用する場合、データベース・バッファ・キャッシュにHugePagesを使用することはできません。HugePagesは、shared_pool、large_poolなど、他の部分のSGAに使用できます。VLM(バッファ・キャッシュ)のためのメモリー割当ては、共有メモリー・ファイル・システム(ramfs/tmpfs/shmfs)を使用して実行されます。メモリー・ファイル・システムでは、HugePagesは予約または使用されません。
- システムの起動後に、HugePagesを割当てまたは解放することはできません。ただし、システム管理者が、使用可能なページ数を変更するか、またはプール・サイズを変更することによって、HugePages構成を変更した場合は例外です。システムの起動時に必要な領域がメモリー内に確保されなかった場合、HugePagesの割当てに失敗します。
- アプリケーションによって余分なHugePagesが使用されない場合にシステムでメモリーが不足することがあるため、HugePagesが適切に構成されていることを確認します。
- インスタンスの起動時に十分なHugePagesがなく、初期化パラメータuse_large_pagesがonlyに設定されている場合、データベースは起動できず、アラート・ログ・メッセージにHugepagesに関して必要な情報が表示されます。

A.7.6 透過的なHugePagesの無効化

インストールを開始する前に、透過的なHugePagesを無効にすることをお薦めします。

透過的なHugePagesメモリーが標準のHugePagesメモリーと異なるのは、カーネルのkhugepagedスレッドが実行時にメモリーを動的に割り当てるためです。標準のHugePagesメモリーは起動時に事前割当てされ実行中には変更されません。

ノート:

透過的な HugePages が UEK2 カーネルおよびそれ以降の UEK カーネルで無効になっていても、Linux システムでは透過的な HugePages をデフォルトで有効にすることができます。

透過的なHugePagesメモリーは、Oracle Linux 6以降、Red Hat Enterprise Linux 6以降、SUSE 11以降のカーネルでデフォルトで有効になります。

透過的なHugePagesを使用すると、実行中にメモリー割当ての遅延が生じます。パフォーマンスの問題を回避するために、透過的なHugePagesはすべてのOracle Databaseサーバーで無効にすることをお薦めします。かわりに標準のHugePagesを使用すると、パフォーマンスが向上します。

透過的なHugePagesが有効かどうかを確認するには、rootユーザーとして次のコマンドのいずれかを実行します。

Red Hat Enterprise Linuxカーネルの場合:

```
# cat /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

その他のカーネルの場合:

```
# cat /sys/kernel/mm/transparent_hugepage/enabled
```

次に示すのは、[always]フラグを有効にして透過的なHugePagesを使用している場合を示す出力例です。

```
[always] never
```

ノート:

透過的な HugePages をカーネルから削除した場合、/sys/kernel/mm/transparent_hugepage と /sys/kernel/mm/redhat_transparent_hugepage のどちらのファイルも存在しません。

透過的なHugePagesを無効にするには:

1. Oracle Linux 7以降およびRed Hat Enterprise Linux 7以降の場合は、/etc/default/grubファイル内でtransparent_hugepage=neverパラメータを追加または変更します。

   ```
   transparent_hugepage=never
   ```

   たとえば:

   ```
   GRUB_TIMEOUT=5
   GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
   GRUB_DEFAULT=saved
   GRUB_DISABLE_SUBMENU=true
   GRUB_TERMINAL_OUTPUT="console"
   GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet numa=off
   transparent_hugepage=never"
   GRUB_DISABLE_RECOVERY="true"
   ```

   ノート:

   このファイル名は、オペレーティング・システムによって異なる場合があります。使用するオペレーティング・システムのドキュメントで正確なファイル名および透過的な HugePages を無効化するステップを確認します。

2. grub2-mkconfig コマンドを実行してgrub.cfgファイルを再生成します。

   ```
   # grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

3. システムを再起動して変更を確定します。

B Oracle SolarisでのOracle Databaseの管理

この付録では、Oracle SolarisでOracle Databaseの管理について説明します。

次の項目が含まれます。

Oracle Solarisの共有メモリー環境

Oracle Solarisシステムでのリソース・プール作成について

マルチCPUバインディング機能について

B.1 Oracle Solarisの共有メモリー環境

この項では、最適化共有メモリー(OSM)、緊密共有メモリー(ISM)、動的緊密共有メモリー(DISM)などの共有メモリー・モデルをOracle Databaseで使用する方法について説明します。

次の項目が含まれます。

- 最適化共有メモリーについて
- 最適化共有メモリーのチェック
- ISMおよびDISMについて
- ISMまたはDISMのチェック
- oradismユーティリティについて
- Oracle DatabaseによるOSM_ ISMおよびDISMの選択方法

B.1.1 最適化共有メモリーについて

12c以降、Oracle Databaseは、Oracle Solaris 10 1/13およびOracle Solaris 11.1 SRU 7.5以上のシステムではOracle Solarisの最適化共有メモリー(OSM)モデルを使用して自動メモリー管理を実装します。

OSMを使用すると、インスタンスを再起動することなく、システム・グローバル領域(SGA)のサイズを動的に変更できます。oradismユーティリティを使用せず、ディスク領域をスワップします。OSMはNUMA用に最適化されています。

B.1.2 最適化共有メモリーのチェック

ノート:

最適化共有メモリーを使用するには、必ず MEMORY_MAX_TARGET を MEMORY_TARGET より大きい値に設定します。

Oracle Solarisで最適化共有メモリー(OSM)が使用されているかどうかを確認するには、次のコマンドを入力します。

```
$ ipcs -dm
```

ALLOC列に整数が表示された場合、OSMが使用されていることを指します。ALLOC列にハイフンが表示された場合、OSMが使用されていないことを指します。

B.1.3 ISMおよびDISMについて

Oracle Solarisシステム上のOracle Databaseでは、Oracleプロセス間で仮想メモリー・リソースを共有するため、共有メモリー・セグメントとして緊密共有メモリー(ISM)が使用されます。ISMを使用すると、共有メモリー・セグメント全体の物理メモリーが自動的にロックされます。

Oracle Solaris 10 1/13およびOracle Solaris 11 SRU 7.5より前のOracle Solaris 10システムでは、動的緊密共有メモリー(DISM)を使用できます。動的緊密共有メモリーを使用すると、Oracle Databaseではセグメントを共有するプロセス間で仮想メモリー・リソースを共有でき、同時にメモリーのページングも可能になります。このため、オペレーティング・システムでは、共有メモリー・セグメント全体の物理メモリーをロックする必要がありません。

B.1.4 ISMまたはDISMのチェック

Oracle Solarisでは、共有メモリーが使用されているかどうかを確認するのに`ipcs -im`コマンドを使用します。たとえば:

```
% ipcs -im
IPC status from <system> as of Thu Aug 19 01:09:30 PDT 2013
T  ID     KEY        MODE       OWNER    GROUP   ISMATTCH
Shared Memory:
m  11  0xacea4150 --rw-rw---- oracle    dba     160
```

`ISMATTCH`フィールドには、この共有メモリー・セグメントにアタッチされた160個のプロセスが表示されます。ただし、`ISMATTCH`では、緊密共有メモリー(ISM)と動的緊密共有メモリー(DISM)が区別されません。

Oracle Solaris 10 1/13およびOracle Solaris 11 SRU 7.5より前のOracle Solaris 10システムでは、ISMまたはDISMが使用されているかどうかを確認したり、どのメモリーのロック・サービスがアクティブかを確認するのに`pmap -xs`コマンドを使用します。たとえば:

```
% ps -ef | grep ora | grep smon
oracle 12524 1 0 05:40:13 ? 0:25 ora_smon_prod
% pmap -xs 12524 | grep ism
0000000380000000 38010880 38010880 - 38010880 256M rwxsR [ ism shmid=0xb ]
0000000C90000000 131072 131072 - 131072 4M rwxsR [ ism shmid=0xb ]
0000000C98000000 16 16 - 16 8K rwxsR [ ism shmid=0xb ]
```

ノート:

`ps -ef`コマンドでは、実行中のバックグラウンド・プロセスがリストされます。この情報は、Oracleデータベース・インスタンスが実行中かどうかを判断する場合に必要です。

`pmap -xs`コマンドの出力には、ISMが使用されていることを意味する`ism`アドレス範囲が3つ表示されます。DISMによってメモリー範囲がロックされる場合、出力には`dism`アドレス範囲が表示されます。

B.1.5 oradismユーティリティについて

Oracle Databaseは、`oradism`ユーティリティを使用して共有メモリーのロックおよびロック解除を実行します。`oradism`ユーティリティは、インストール時に自動的に設定されます。このため、動的SGAを使用するための構成作業は不要です。

`oradism`ユーティリティのプロセス名は、`ora_dism_sid`です。`sid`はシステム識別子です。DISMを使用している場合、このプロセスはインスタンスの起動時に開始され、インスタンスの停止時に自動的に終了します。

`oradism`ユーティリティが正しく設定されていないことを示すメッセージがアラート・ログに表示された場合は、`oradism`ユーティ

リティが$ORACLE_HOME/binディレクトリにあり、スーパーユーザーの権限が付与されていることを確認してください。

ノート:

最適化共有メモリー(OSM)では、oradism ユーティリティを使用しません。

B.1.6 Oracle DatabaseによるOSM, ISMおよびDISMの選択方法

最適化共有メモリー(OSM)が使用できるOracle Solarisシステムの場合、Oracle DatabaseではOSMが自動的に使用されます。OSMの詳細は、「最適化共有メモリーについて」を参照してください。

OSMが使用できないシステムの場合、Oracle Databaseでは、次の基準に基づいて緊密共有メモリー(ISM)または動的緊密共有メモリー(DISM)が自動的に選択されます。

- システムでDISMを使用でき、SGA_MAX_SIZE初期化パラメータの値が、結合されたすべてのSGAコンポーネントに必要なサイズよりも大きい場合に、DISMが使用されます。これにより、Oracle Databaseでは、使用される物理メモリー量のみがロックされます。
- 起動時に共有メモリー・セグメント全体が使用中の場合、またはSGA_MAX_SIZEパラメータの値が、結合されたすべてのSGAコンポーネントに必要なサイズ以下の場合に、ISMが使用されます。

Oracle Databaseでは、ISMまたはDISMのいずれを使用するかに関係なく、インスタンスの起動後は常に、動的にサイズ変更できるコンポーネント(バッファ・キャッシュなど)、共有プールおよびラージ・プール間でメモリーを交換できます。Oracle Databaseでは、動的SGAコンポーネントからメモリーを解放し、それを別のコンポーネントに割り当てることができます。

DISMの使用時は、共有メモリー・セグメントがメモリー内で暗黙的にロックされないため、Oracle Databaseは起動時に使用している共有メモリーを明示的にロックします。動的なSGA操作によって使用される共有メモリーが増えると、Oracle Databaseは使用中のメモリーを明示的にロックします。動的なSGA操作によって共有メモリーが解放されると、Oracle Databaseは解放されたメモリーを明示的にロック解除するので、解放されたメモリーは他のアプリケーションで使用できるようになります。

ノート:

サーバー・パラメータ・ファイルの LOCK_SGA パラメータは、TRUE に設定しないでください。設定した場合、Oracle Database 12c は起動できません。

B.2 Oracle Solarisのリソース・プールの作成について

Oracle Solarisのリソース・プールは、一連の専用CPUをデータベース・インスタンスに関連付けることによってデータベースのパフォーマンスを向上させます。各データベース・インスタンスはそのリソース・プールのリソースのみを使用します。

大規模サーバーで統合を行う場合、CPUとメモリーの特定のサブセットにデータベースを制限することをお薦めします。この機能により、Oracle DatabaseインスタンスのCPUとメモリーの制約を簡単に有効にすることができます。

setup_resource_pool.shスクリプトを使用して、Oracle Solarisのリソース・プールを作成します。このスクリプトは、My Oracle Support Webサイトのノート1928328.1からダウンロードします。

https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1928328.1

B.3 マルチCPUバインディング機能について

マルチCPUバインディング(MCB)機能は、パフォーマンス改善のためにリソース管理ポリシーの一部として使用できます。

マルチCPUバインディング(MCB)は1つのプロジェクトを特定のCPUセットにバインドするOracle Solarisプロジェクト・リソース管理の機能ですが、排他的にCPUをバインドすることはありません。MCBでは、他のプロセスでもこれらのCPUを使用でき、パーティションの重複が許可されています。MCBはOracle Solaris 11.3以降でサポートされています。

リソース・プール機能でも、CPUのバインドが可能です。ただし、このメソッドでは、システムのプロセッサをハード・パーティショニングする必要があります。リソース・プールではパーティションは重複できません。

MCBはOracle Solarisプロジェクトを介して割当て、変更、削除が可能です。標準のコマンドライン・ツール`projadd(1M)`および`projmod(1M)`を使用して、プロジェクト・ファイルを作成または変更します。

データベース・ユーザーの場合、リソース・プールでMCBを使用する利点は次のとおりです。

- 特定のI/Oまたはネットワーク・デバイスなどの近くの特定のNUMAの場所に、Oracleインスタンスをバインドできる機能。
- システムをパーティショニングする権限を持つ管理者がいなくても、複数のOracleインスタンスをホスト上の様々なコアまたはソケットにバインドし、パフォーマンスの分離を促進できる機能。

関連項目:

「*Oracle Solaris 11.3*でのリソース管理」のプロジェクトを使用したマルチCPUバインディングの割当て、変更、および削除に関する項およびマルチCPUバインディングの割当て、変更、および削除のためのプロジェクトの使用方法に関する項。

C IBM AIX on POWER Systems (64-Bit)での Oracle Databaseの管理

この付録では、IBM AIX on POWER Systems (64-bit)でのOracle Databaseの管理について説明します。

次のトピックが含まれています:

- メモリーおよびページング
- ディスク入出力の問題
- CPUのスケジューリングおよびプロセスの優先順位
- AIXTHREAD_SCOPE環境変数
- ネットワーク情報サービスの外部ネーミングのサポート
- Oracle JDBC Thinドライバを使用したIBM Java Secure Socket Extension Providerの構成

C.1 メモリーおよびページング

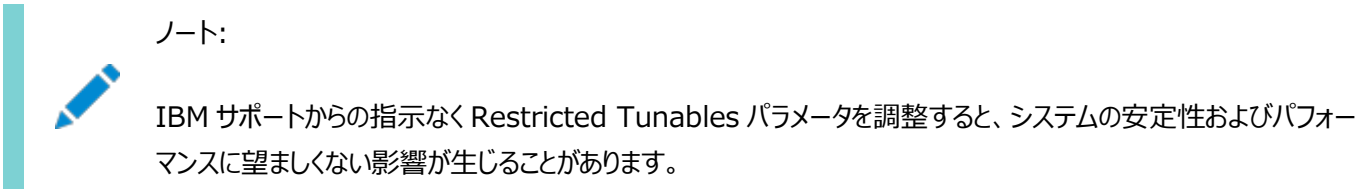
メモリーの競合は、プロセスに必要なメモリー量が、使用できる容量よりも大きくなったときに発生します。このようなメモリー不足に対処するため、メモリーとディスク間でプログラムおよびデータのページングが行われます。

この項では、次の項目について説明します。

- カーネル・パラメータ
- 十分なページング領域の割当て
- ページングの制御
- データベース・ブロック・サイズの設定
- ログ・アーカイブ・バッファのチューニング
- 入出力バッファおよびSQL*Loader

C.1.1 カーネル・パラメータ

デフォルトのAIXカーネル設定を使用することをお薦めします。IBMサポートのみで適切に推奨されているようにカーネル設定を調整する必要があります。



ノート:

IBM サポートからの指示なく Restricted Tunables パラメータを調整すると、システムの安定性およびパフォーマンスに望ましくない影響が生じることがあります。

C.1.2 十分なページング領域の割当て

通常、ページング領域(スワップ領域)が十分に割り当てられていないと、システムのレスポンスが停止したり、レスポンス時間が非常に遅くなります。IBM AIX on POWER Systems (64-bit)では、RAWディスク・パーティションにページング領域を動

的に追加できます。構成するページング領域の大きさは、実装されている物理メモリーの量およびアプリケーションのページング領域要件によって異なります。ページング領域の使用量を監視するには、lspsコマンドを使用します。システムのページング・アクティビティを監視するには、vmstatコマンドを使用します。ページング領域を増やすには、smit pgspコマンドを使用します。

ページング領域を事前に割り当てる場合は、ページング領域をRAMの量よりも大きい値に設定することをお薦めします。ただし、IBM AIX on POWER Systems (64-bit)では、ページング領域は必要になるまで割り当てられません。システムでは、実メモリーが不足した場合にのみスワップ領域が使用されます。メモリーのサイズを正しく設定した場合は、ページングが行われないため、ページング領域を小さくできます。要求されるページ数が大きく変動しないワークロードの場合は、小さいページング領域でも適切に動作します。ページングが極端に増加する可能性があるワークロードの場合は、最大ページ数を処理できるだけの十分なページング領域が必要です。

原則として、ページング領域の初期設定はRAMの半分のサイズに4GBを追加したものになります(最大は単一の内部ディスクのサイズまで)。lsps -aコマンドを使用してページング領域の使用量を監視し、vmstatコマンドを使用してシステムのページング・アクティビティを監視します。lsps -aにより出力されるメトリック%Usedは、正常なシステムでは通常25%未満です。過剰な量のスワッピングはパフォーマンスに重大な影響を与えるため、適切にサイズ設定されているデプロイメントにはページング領域はほとんど必要ありません。ページング領域およびスワッピングを過剰に使用すると、システム上のRAMが不足することがあります。

注意:

ページング領域のサイズは小さくしないでください。小さくすると、領域が不足してアクティブなプロセスが終了します。一方、ページング領域のサイズが大きすぎても、悪影響はほとんどありません。

Oracleドキュメントでは、Oracle Databaseの初期設定として次の値が推奨されています。

RAM	スワップ領域
1 から 2GB	RAM のサイズの 1.5 倍
2GB から 16GB	RAM のサイズと同じ
16GB 超	16GB

RAMおよびOracle Grid Infrastructureのスワップ領域値は次のとおりです。

- 4GB RAMから16GB RAMの間。スワップ領域はRAMのサイズと同じである必要があります。
- 16GB RAMを超える場合、スワップ領域は16GBと同じである必要があります。

個々のサーバー環境は異なるため、Oracle Database 19c環境では19cのメモリー・フットプリントおよび4KBから64KBまでのページ・サイズの増大に基づいて一部の追加メモリーが保証されています。ワークロードをリバランスしてページングを減らす必要がある場合がありますが、これによりシステムのパフォーマンスに影響があります。

C.1.3 ページングの制御

過剰なページングが頻繁に発生する場合は、実メモリーが過剰にコミットされていることを示します。通常は、次のように対処します。

- ページングが頻繁に発生しないようにします。ただし、システムに超高速の拡張記憶域を装備し、メモリーと拡張記憶域間のページングが、Oracle DatabaseによるSGAとディスク間のデータの読取り/書込みよりも大幅に速くなる場合

を除きます。

- 制限されたメモリー・リソースを、システム・パフォーマンスが最も向上する場所に割り当てます。場合によっては、メモリー・リソース要件とその影響のバランスを取るために、この処理を繰り返し行う必要があります。
- メモリーが不足している場合は、システム内のメモリーを必要とするプロセスおよび要素を優先順に並べたリストを作成します。パフォーマンスが最も向上する場所に、メモリーを割り当てます。優先順リストの例を次に示します。
    1. オペレーティング・システムおよびRDBMSカーネル(SGAおよびそのコンポーネント、バッファ、キャッシュおよび共有プールを含めるため)
    2. ユーザー・プロセスおよびアプリケーション・プロセス

たとえば、Oracle Databaseの動的パフォーマンス表およびビューを問い合せた結果、共有プールとデータベース・バッファ・キャッシュの両方にメモリーを追加する必要があるとします。この場合、制限された予備メモリーは、データベース・ブロック・バッファ・キャッシュではなく共有プールに割り当てるとパフォーマンスが向上します。これらの選択肢は、データベースのロードの性質または形状によって異なります。

次のIBM AIX on POWER Systems (64-bit)コマンドを実行すると、ページングのステータスおよび統計が表示されます。

- `vmstat -s`
- `vmstat interval [repeats]`
- `sar -r interval [repeats]`

C.1.4 データベース・ブロック・サイズの設定

Oracle Databaseのブロック・サイズを構成すると、入出力スループットを改善できます。IBM AIX on POWER Systems (64-bit)では、DB_BLOCK_SIZE初期化パラメータの値を2から32KBに設定できます。デフォルト値は4KBです。Oracle Databaseがジャーナル・ファイル・システムにインストールされている場合は、そのブロック・サイズをファイル・システムのブロック・サイズ(JFSでは4KB、IBM Spectrum Scale (GPFS)では16KBから1MB)の倍数にする必要があります。データベースがRAWパーティション上にある場合は、Oracle Databaseのブロック・サイズをオペレーティング・システムの物理ブロック・サイズ(IBM AIX on POWER Systems (64-Bit)では512バイト)の倍数にします。

Oracle Databaseのブロック・サイズは、オンライン・トランザクション処理ワークロードまたは複合ワークロードの環境では小さめ(2または4KB)に設定し、意思決定支援システム・ワークロードの環境では大きめ(8、16または32KB)に設定することをお薦めします。

C.1.5 ログ・アーカイブ・バッファのチューニング

トランザクションが長い場合や数が多い場合は特に、LOG_BUFFERサイズを大きくすることで、データベースのアーカイブ速度を向上させることができます。ログ・ファイル入出力アクティビティおよびシステム・スループットを監視して、最適なLOG_BUFFERサイズを決定します。LOG_BUFFERパラメータをチューニングするときは、通常のデータベース・アクティビティの全体的なパフォーマンスが低下しないように注意します。

パフォーマンスを向上させるため、デフォルトの4KBではなく512バイトのagblksizeで、REDOログ・ファイルと制御ファイル用に個別のファイル・システム(または両方用に単一のファイル・システム)を作成します。

C.1.6 入出力バッファおよびSQL*Loader

SQL*Loaderダイレクト・パス・オプションを使用しながらデータを並行してロードするなど、データを高速にロードする場合は、CPU時間の大半が入出力完了の待機時間として使用されます。バッファ数を増やすことで、CPU使用率を最大化し、スルー

プット全体を向上させることができます。

バッファ数(SQL*LoaderのBUFFERSパラメータで設定)は、使用可能なメモリーの量およびCPU使用率を最大化する程度によって異なります。

パフォーマンスの向上は、CPU使用率およびデータのロード時に使用する並列度によって変化します。

C.2 ディスク入出力の問題

ディスク入出力の競合は、メモリー管理が良好でない(その結果としてページングおよびスワッピングが発生する)場合や、ディスク間の表領域およびファイルの分散が適切でない場合に発生します。

`filemon`、`sar`、`iostat`などのIBM AIX on POWER Systems (64-bit)ユーティリティおよびその他のパフォーマンス・ツールを使用して入出力アクティビティの高いディスクを特定し、複数のディスク・ドライブに入出力アクティビティを均等に分散します。

この項では、次の項目について説明します。

- IBM AIX on POWER Systems (64-Bit)論理ボリューム・マネージャ
- RAW論理ボリュームと比較した場合のジャーナル・ファイル・システムの使用
- 非同期入出力の使用
- 入出力スレーブ
- DB_FILE_MULTIBLOCK_READ_COUNTパラメータの使用
- ディスク入出力ペーシングのチューニング
- Oracle Databaseによるミラー復元

C.2.1 IBM AIX on POWER Systems (64-Bit)論理ボリューム・マネージャ

IBM AIX on POWER Systems (64-bit)論理ボリューム・マネージャは、複数のディスク間にデータをストライプ化して、ディスクの競合を減らすことができます。ストライプ化の主な目的は、大容量の順次ファイルに対する読取り/書込みのパフォーマンスを向上させることです。記憶域サブシステムが向上したため、LVMのストライプ化の使用は推奨されなくなりました。記憶域サブシステムによるデフォルトのストライプ化の使用をお薦めします。AIXパーティションに存在するLUNは論理的であって物理的ではないため、オペレーティング・システムでデータの物理的な場所が認識されなくなりました。

C.2.2 RAW論理ボリュームと比較した場合のジャーナル・ファイル・システムの使用

ジャーナル・ファイル・システムまたはRAW論理ボリュームのどちらを使用するかを決定するときには、次のことを考慮してください。

- ファイル・システムは、実装の多様化に伴い、継続的に改善されています。
- 様々なベンダーが、各ディスクの長所を生かすために、様々な方法でファイル・システム・レイヤーを実装しています。その結果、プラットフォーム間でファイル・システムを比較することが難しくなっています。
- IBM AIX on POWER Systems (64-bit)に組み込まれているダイレクト入出力機能および同時入出力機能により、ファイル・システムのパフォーマンスはRAW論理ボリュームと同じレベルまで向上します。
- 以前のバージョンのIBM AIX on POWER Systems (64-bit)では、ファイル・システムはバッファに対する読取り/書込みのみをサポートしており、inodeロックが不完全なために余計な競合が発生していました。この2つの問題は、JFS2の同時入出力機能およびSpectrum Scale (GPFS)のダイレクト入出力機能によって解決されています。

- より強力な論理ボリューム・マネージャ・インタフェースを導入すると、RAW論理ボリュームに基づいた論理ディスクの構成およびバックアップの作業が大幅に減少します。
- Oracle ASMは、RAWディスク・デバイスがディスク・グループに追加された場合に最も効果的に機能します。Oracle ASMを使用している場合は、論理ボリューム・マネージャを使用してストライプ化を行わないでください。Oracle ASMは、ストライプ化およびミラー化を実装します。

ノート:

Oracle RAC オプションを使用するには、Oracle ASM ディスク・グループまたは Spectrum Scale (GPFS)ファイル・システムにデータファイルを配置する必要があります。JFS または JFS2 は使用できません。Spectrum Scale (GPFS)を使用すると、ダイレクト入出力が暗黙的に有効になります。

ファイル・システム・オプション

IBM AIX on POWER Systems (64-bit)では、ダイレクト入出力および同時入出力がサポートされています。ダイレクト入出力および同時入出力のサポートにより、データベース・ファイルがファイル・システム上に存在できるようになります。これは、Oracle Databaseが提供する機能を使用して、オペレーティング・システムのバッファ・キャッシュを回避し、冗長なinodeロック操作を排除することで実現されます。

次の表に、IBM AIX on POWER Systems (64-bit)で使用できるファイル・システムとその推奨設定を示します。

ファイル・システム	オプション	説明
JFS	dio	JFS では、同時入出力は使用できません。ダイレクト入出力は使用できますが、同時入手力を使用した JFS2 と比較してパフォーマンスが劣ります。
JFS ラージ・ファイル	なし	128KB の位置合せ制約によってダイレクト入出力が使用できないため、JFS ラージ・ファイルを Oracle Database に使用することは推奨されません。
JFS2	cio	同時入出力は、同一のファイルに対して複数の同時リーダー/ライターをサポートしているため、ダイレクト入出力よりも JFS2 に適した設定です。ただし、JFS2/CIO に対する IBM AIX on POWER Systems (64-bit) 制限のため、同時入出力は Oracle データファイル、制御ファイルおよびログ・ファイルでのみ使用されます。このような目的専用のファイル・システムにのみ適用してください。同じ理由から、`CIO` オプションでマウントする JFS2 ファイル・システムでは、Oracle ホーム・ディレクトリはサポートされません。たとえば、インストール時に、`CIO` オプションでマウントする JFS2 ファイル・システムに Oracle ホーム・ディレクトリを配置するように誤って指定した場合は、Oracle に再リンクしようとすると、次のエラーが表示されることがあります。 `"ld: 0711-866 INTERNAL ERROR: Output symbol table size miscalculated"`

ファイル・システム	オプション	説明
		ノート: Oracle Database 11g リリース 2 (11.2.0.2)以上の場合、IBM AIX on POWER Systems (64-bit) 6.1 以上のシステムでは、JFS2 ファイル・システム上で CIO マウント・オプションを使用しないことをお薦めします。最新の Oracle Database リリースの場合は、Oracle の内部で O_CIOR オプションでファイルシステムが開かれるため、CIO マウント・オプションを使用する必要はありません。これにより、他のアプリケーションで Oracle データ・ファイルを O_CIO オプションなしで読取り専用モードで開くことができるとともに、CIO のメリットを受けられます。
Spectrum Scale (GPFS)	NA	Oracle Database は、最適なパフォーマンスを得るために、Spectrum Scale に対してダイレクト入出力を暗黙的に有効にします。Spectrum Scale のダイレクト入出力は、すでに複数のノード上の複数のリーダー/ライターをサポートしています。したがって、Spectrum Scale では、ダイレクト入出力と同時入出力は同じです。

JFSおよびJFS2の考慮事項

JFS2ファイル・システムにOracle Databaseログを配置している場合、構成を最適化するには、agblksize=512オプションを使用してファイル・システムを作成し、CIOオプションでマウントします。

Oracle Database 12cより前のリリースでは、JFS/JFS2において、ダイレクト入出力および同時入出力をファイル・レベルで有効にできませんでした。したがって、最適なパフォーマンスを得るために、Oracleホーム・ディレクトリおよびデータファイルを別個のファイル・システムに配置する必要がありました。つまり、Oracleホーム・ディレクトリをデフォルト・オプションでマウントしたファイル・システムに配置し、データファイルおよびログをDIOまたはCIOオプションを使用してマウントしたファイル・システムに配置していました。

Oracle Database 12cでは、JFS/JFS2において、ダイレクト入出力および同時入出力をファイル・レベルで有効にできます。そのためには、サーバー・パラメータ・ファイルのFILESYSTEMIO_OPTIONSパラメータをSETALLまたはDIRECTIOに設定します。これにより、すべてのデータファイル入出力に対して、JFS2での同時入力およびJFSでのダイレクト入出力が有効になります。これは、DIRECTIO設定により、通常は使用しない非同期入出力が無効になるためです。この12cの機能により、Oracleホーム・ディレクトリと同じJFS/JFS2ファイル・システムにデータファイルを配置し、ダイレクト入出力または同時入出力を使用してパフォーマンスを向上させることができます。前述のように、最適なパフォーマンスを得るためには、Oracle Databaseログを別個のJFS2ファイル・システムに配置する必要があります。

関連項目:

詳細は、『Oracle Architecture and Tuning on AIX v2.30』を参照してください。

Spectrum Scaleの考慮事項

Spectrum Scale (GPFS)を使用している場合、すべての目的に同じファイル・システムを使用できます。Oracleホーム・ディレクトリとしての使用や、データ・ファイルやログの格納などです。最適なパフォーマンスを得るため、大きなSpectrum Scaleブロック・サイズ(通常、512KB以上)を使用する必要があります。Spectrum Scaleはスケーラビリティのために設計されているため、データの量が単一のSpectrum Scaleファイル・システムに収まるかぎり、複数のSpectrum Scaleファイル・システムを作成する必要はありません。

C.2.3 非同期入出力の使用

Oracle Databaseでは、IBM AIX on POWER Systems (64-bit)が提供する非同期入出力を最大限に利用して、データベース・アクセスを高速化しています。

IBM AIX on POWER Systems (64-bit)では、ファイル・システム・パーティション上に作成されたデータベース・ファイルに対して、非同期入出力がサポートされています。ファイル・システムに対して非同期入出力を使用するときは、リクエストがキューから取り出されてから完了するまで、カーネル・データベース・プロセス(aioserver)が各リクエストを制御します。aioserverサーバーの数により、システムで同時に処理できる非同期入出力リクエストの数が決まります。AIOチューナブルのデフォルト設定が大幅に増大されているため、AIOチューナブルを調整する必要はありません。

C.2.4 入出力スレーブ

入出力スレーブは、入出力のみを実行する特殊なOracleプロセスです。非同期入出力がデフォルトであり、IBM AIX on POWER Systems (64-bit)でのOracleによる入出力操作の実行に推奨されている方法であるため、入出力スレーブはIBM AIX on POWER Systems (64-bit)ではほとんど使用されません。入出力スレーブは、共有メモリー・バッファから割り当てられます。入出力スレーブには、次の表に示す初期化パラメータを使用します。

パラメータ	値の範囲	デフォルト値
DISK_ASYNCH_IO	true/false	true
TAPE_ASYNCH_IO	true/false	true
BACKUP_TAPE_IO_SLAVES	true/false	false
DBWR_IO_SLAVES	0 - 999	0
DB_WRITER_PROCESSES	1-20	1

通常、この表のパラメータは調整しません。ただし、ワークロードが大きい場合に、データベース・ライターがボトルネックになることがあります。その場合は、DB_WRITER_PROCESSESの値を大きくします。このデータベース・ライター・プロセスの数は、システムまたはパーティション内のCPUのペアにつき1つですが、原則として、この数は増やさないでください。

非同期I/Oの無効化が必要な場合があります。たとえば、デバッグのためにOracleサポートから指示された場合などです。DISK_ASYNCH_IOおよびTAPE_ASYNCH_IOパラメータを使用すると、ディスクまたはテープ・デバイスに対する非同期I/Oを無効にできます。TAPE_ASYNCH_IOのサポートは、メディア・マネージャ・ソフトウェアによってサポートされている場合にのみ使用可能であり、Recovery Managerの場合は、BACKUP_TAPE_IO_SLAVESがtrueの場合のみ使用可能です。

DBWR_IO_SLAVESパラメータは、DISK_ASYNCH_IOパラメータがfalseに設定されている場合にのみ、0(ゼロ)より大きい値に設定します。設定しないと、データベース・ライター・プロセスがボトルネックになります。この場合、IBM AIX on POWER Systems (64-bit)でのDBWR_IO_SLAVESパラメータの最適値は4です。

C.2.5 DB_FILE_MULTIBLOCK_READ_COUNTパラメータの使用

Oracle Database 19cでダイレクト入出力または同時入出力を使用している場合、IBM AIX on POWER Systems (64-bit)ファイル・システムでは、順次スキャンでの先読みが実行されません。このため、ダイレクト入出力または同時入出力が

Oracleデータファイルに対して有効になっている場合、サーバー・パラメータ・ファイルのDB_FILE_MULTIBLOCK_READ_COUNT値を大きくする必要があります。DB_FILE_MULTIBLOCK_READ_COUNT初期化パラメータで指定されているように、Oracle Databaseによって先読みが実行されます。

DB_FILE_MULTIBLOCK_READ_COUNT初期化パラメータに大きい値を設定すると、通常は順次スキャンでの入出力スループットが向上します。IBM AIX on POWER Systems (64-bit)では、このパラメータの範囲は1から512ですが、16より大きい値を使用しても、通常はそれ以上のパフォーマンス効果は得られません。

このパラメータは、DB_BLOCK_SIZEパラメータの値との積が論理ボリューム・マネージャのストライプ・サイズよりも大きくなるように設定します。このような設定により、使用できるディスクが増加します。

C.2.6 ディスク入出力ペーシングのチューニング

ディスク入出力ペーシングとは、システム管理者がファイルに対して保留中の入出力リクエストの数を制限できるようにするIBM AIX on POWER Systems (64-bit)のメカニズムです。このメカニズムにより、ディスク入出力が頻繁に発生するプロセスでCPUが飽和状態になるのを防ぐことができます。このため、対話型プロセスやCPU使用量の多いプロセスのレスポンス時間に遅延が発生しません。

ディスク入出力ペーシングを行うには、最高水位標および最低水位標という2つのシステム・パラメータを調整します。保留中の入出力リクエストがすでに最高水位標に達しているファイルに対してプロセスが書込みを実行すると、プロセスはスリープ状態になります。未処理の入出力リクエストの数が最低水位標以下になると、プロセスはスリープ状態から解放されます。

IBM AIX 6.1バージョンから、ファイル・システムおよびAIX I/Oサブシステムはファイル・システムI/Oの大きな増加に対応するように変更されています。これらの変更に加えて、I/Oペーシングのデフォルト値も変更されました。IBM AIX 6.1バージョン以下のデフォルト値は0 (ゼロ)からI/Oペーシングなしです。IBM AIX 6.1バージョン以上のデフォルト値は次のとおりです。

- `minpout=4096`
- `maxpout=8193`

C.2.7 Oracle Databaseによるミラー復元

RAW論理ボリュームに割り当てられたOracleデータファイルに対してミラー書込み整合性を無効にすると、システム障害後のOracle Databaseのクラッシュ・リカバリ・プロセスでは、ミラー復元を使用したリカバリが行われます。このミラー復元プロセスにより、データベースの不整合または破損を防ぐことができます。

クラッシュ・リカバリ時に、論理ボリュームに割り当てられたデータファイルに複数のコピーがある場合、ミラー復元プロセスでは、すべてのコピーのデータ・ブロックに対してチェックサムを実行します。その後、次のいずれかの処理を実行します。

- コピー内のデータ・ブロックのチェックサムが有効である場合、そのコピーを使用してチェックサムが無効なコピーを更新します。
- すべてのコピーでブロックのチェックサムが無効である場合、REDOログ・ファイルの情報を使用してブロックを再構築します。次に、データファイルを論理ボリュームに書き込み、すべてのコピーを更新します。

IBM AIX on POWER Systems (64-bit)では、ミラー復元プロセスは、RAW論理ボリュームに割り当てられたデータファイルのうち、ミラー書込み整合性が無効になっているデータファイルに対してのみ機能します。ミラー化論理ボリューム上のデータファイルのうち、ミラー書込み整合性が有効になっているデータファイルには、ミラー書込み整合性によってすべてのコピーの同期が保証されているため、ミラー復元は必要ありません。

以前のリリースのOracle Databaseのアップグレード中にシステムに障害が発生し、論理ボリューム上のデータファイルのミラー書込み整合性が無効になっていた場合は、syncvgコマンドを実行してミラー化論理ボリュームを同期化してからOracle

Databaseを起動してください。ミラー化論理ボリュームを同期化せずにデータベースを起動すると、論理ボリュームのコピーから誤ったデータを読み取る場合があります。

ノート:

ディスク・ドライブに障害が発生した場合、ミラー復元は行われません。syncvg コマンドを実行してから、論理ボリュームを再度アクティブにする必要があります。

注意:

ミラー復元は、データファイルに対してのみサポートされています。REDO ログ・ファイルのミラー書込み整合性は無効にしないでください。

C.3 CPUのスケジューリングおよびプロセスの優先順位

CPUも、プロセスの競合が発生する可能性があるシステム・コンポーネントです。ほとんどの場合、IBM AIX on POWER Systems (64-bit)カーネルによってCPUは効果的に割り当てられますが、プロセスの多くはCPUサイクルをめぐって競合します。システムに複数のCPU(SMP)が搭載されている場合は、各CPUで様々なレベルの競合が発生する可能性があります。

C.4 AIXTHREAD_SCOPE環境変数

IBM AIX 7.1バージョンを使用し、IBM AIX 6.1のデフォルト(システム全体)としてAIXTHREAD_SCOPE環境変数をS(1:1)に設定することをお薦めします。

関連項目:

スレッド・チューニングの詳細は、『Thread tuning』を参照してください。

C.5 ネットワーク情報サービスの外部ネーミングのサポート

IBM AIX on POWER Systems (64-bit)では、ネットワーク情報サービスの外部ネーミング・アダプタがサポートされています。ネットワーク情報サービスの外部ネーミングを構成および使用する方法の詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。

C.6 Oracle JDBC Thin Driverを使用したIBM Java Secure Socket Extensionプロバイダの構成

IBM Java 1.6 SR 16はOracle Database 12cリリース1(12.1)に同梱されています。IBM JDKにSSLを構成する場合は、次の問題が発生する場合があります。

- IBM JSSEがSSLv2Hello SSLプロトコルをサポートしないただし、クライアントのカプセル化からのSSLv2Helloメッセージを受け入れます。

  SSLv3またはTLS1.0ハロー・メッセージ。

POODLEのセキュリティの問題の後、SSLv3は推奨されなくなったため、Thin JDBCコネクタを使用するSSLクライアントでは、oracle.net.ssl_versionシステム・プロパティをTLSv1 SSLプロトコルまたはSSLv3 SSLプロトコルを選択するように設定する必要があります。システム・プロパティの推奨設定は、推奨されるSSLv3上でTLSV1.0、TLSV1.1およびTLSV1.2を設定することです。それ以外の場合、接続に失敗します。

- IBM JSSEは匿名暗号を許可しない

  匿名暗号を使用するSSLクライアントでは、デフォルトのトラスト・マネージャを、匿名暗号を許可するカスタム・トラスト・マネージャに置き換える必要があります。

関連項目:

- 詳細は、『Padding Oracle On Downgraded Legacy Encryption (POODLE) security vulnerability』を参照してください。
- カスタム・トラスト・マネージャの作成およびインストールの詳細は、IBM JSSEのドキュメントを参照してください。

D HP-UXでのOracle Databaseの管理

この付録では、HP-UXシステムでOracle Databaseを管理する方法について説明します。

次の項目が含まれます。

- Oracleインスタンス用のHP-UX共有メモリー・セグメント
- HP-UX SCHED_NOAGEスケジュール・ポリシー
- 軽量タイマーの実装
- 非同期入出力
- 大規模メモリーの割当てとOracle Databaseのチューニング
- CPU_COUNT初期化パラメータおよびHP-UX動的プロセッサ再構成
- ネットワーク情報サービスの外部ネーミングのサポート
- 拡張ホスト名および拡張ノード名のアクティブ化と設定

D.1 Oracleインスタンス用のHP-UX共有メモリー・セグメント

Oracle Databaseは、インスタンスの起動時に、Oracleシステム・グローバル領域(SGA)の作成用に割り当てられた共有メモリーをHP-UX shmmaxカーネル・パラメータの値で除算して、メモリー・セグメントを作成します。たとえば、1つのOracleインスタンスに割り当てられた共有メモリーが64GBで、shmmaxパラメータの値が1GBの場合、Oracle Databaseはそのインスタンスに対して64個の共有メモリー・セグメントを作成します。

1つのOracleインスタンスに対して複数の共有メモリー・セグメントが作成されると、パフォーマンスが低下する可能性があります。これは、Oracle Databaseでインスタンスが作成されるときに、各共有メモリー・セグメントが一意の保護キーを受け取るためです。HP-UX Itaniumのシステム・アーキテクチャで使用できる保護キーの数は14です。

Oracleインスタンスが作成する共有メモリー・セグメントが保護キーの数より多い場合、HP-UXオペレーティング・システムは、保護キー・フォルトを表示します。

shmmaxパラメータ値は、システムで使用できる物理メモリーの量に設定することをお薦めします。この設定により、1つのOracleインスタンスの共有メモリー全体が1つの共有メモリー・セグメントに割り当てられ、インスタンスに必要な保護キーが1つのみになります。

システムのアクティブな共有メモリー・セグメントのリストを表示するには、次のコマンドを実行します。

```
$ ipcs -m
```

Oracle Databaseがインスタンスに対して保護キーの数よりも多いセグメントを作成する場合は、shmmaxカーネル・パラメータの値を大きくします。

関連項目:

カーネル・パラメータの推奨最小値の詳細は、『Oracle Databaseインストレーション・ガイド』を参照してください。

D.2 HP-UX SCHED_NOAGEスケジュール・ポリシー

HP-UXでは、ほとんどのプロセスがタイムシェアリング・スケジュール・ポリシーを使用します。タイムシェアリングが適用されると、重要な操作(ラッチの保持など)が実行されるときにOracleプロセスがスケジュールから除外されるため、パフォーマンスが低下する場合があります。HP-UXには、特にこの問題に対処した修正済のスケジュール・ポリシーであるSCHED_NOAGEが用意されています。通常のタイムシェアリング・ポリシーとは異なり、SCHED_NOAGEを使用してスケジュールが設定されたプロセスは、優先順位が上下したり、優先使用されることがありません。

この機能は、オンライン・トランザクション処理環境に適しています。これは、オンライン・トランザクション処理環境では重要なリソースをめぐって競合が発生することがあるためです。Oracle DatabaseでSCHED_NOAGEポリシーを使用すると、オンライン・トランザクション処理環境では、パフォーマンスが10%以上も向上する可能性があります。

SCHED_NOAGEポリシーを意思決定支援環境で使用しても、同程度のパフォーマンス効果は得られません。これは、リソースの競合がほとんど発生しないためです。アプリケーションおよびサーバーの環境はそれぞれ異なるため、使用している環境に対してSCHED_NOAGEポリシーが有効であることをテストして検証する必要があります。SCHED_NOAGEを使用する場合、Oracleプロセスに最も高い優先順位を割り当てる際には注意が必要です。SCHED_NOAGEの最も高い優先順位をOracleプロセスに割り当てると、システムのCPUリソースを使い果し、他のユーザー・プロセスのレスポンスが停止する可能性があります。

D.2.1 Oracle Database用のSCHED_NOAGEの有効化

Oracle DatabaseでSCHED_NOAGEスケジュール・ポリシーを使用できるようにするには、OSDBAグループ(通常はdbaグループ)に、スケジュール・ポリシーの変更およびOracleプロセスの優先順位レベルの設定を行うためのRTSCHEDおよびRTPRIO権限が必要です。dbaグループにこれらの権限を付与するには:

1. rootユーザーでログインします。
2. テキスト・エディタを使用して/etc/privgroupファイルを開くか、必要な場合は作成します。
3. OSDBAグループの名前で始まる次の行を追加または編集し、システムが再起動するたびにこのグループに付与するRTPRIOおよびRTSCHED権限を指定します。

   ```
   dba RTPRIO RTSCHED
   ```

4. ファイルを保存してテキスト・エディタを終了します。
5. 次のコマンドを入力し、OSDBAグループに権限を付与します。

   ```
   # /usr/sbin/setprivgrp -f /etc/privgroup
   ```

6. 次のコマンドを入力し、権限が正しく設定されていることを確認します。

   ```
   # /usr/bin/getprivgrp dba
   ```

HPUX_SCHED_NOAGE初期化パラメータを各インスタンスのパラメータ・ファイルに追加し、このパラメータにプロセスの優先順位レベルを整数値で設定します。サポートされる値の範囲は178から255です。値が小さくなるほど優先順位が高くなります。すべてのOracleプロセスについて、HPUX_SCHED_NOAGE初期化パラメータのデフォルト値は178です。LMSプロセスでは、_os_sched_high_priority初期化パラメータを設定すると、この値を変更できます。_os_sched_high_priorityパラメータが31から2の間の場合、LMSプロセスはSCHED_RRで設定された優先順位で実行されます。パラメータの値が178から255の間の場合、プロセスはSCHED_NOAGEで設定された値で実行されます。ただし、LMSは、HPUX_SCHED_NOAGEの値より低い優先順位レベルでは実行されません。

HPUX_SCHED_NOAGEパラメータの設定が範囲外である場合、Oracle Databaseは自動的にそのパラメータを許容値に設定し、新しい値が設定されたSCHED_NOAGEポリシーを使用して処理を続行します。また、新しい設定に関するメッセージを

alert_sid.logファイルに生成します。ユーザーまたは自動再調整によってOracleプロセスに最も高い優先順位が割り当てられた場合は常に、Oracle Databaseは、システムのCPUリソースが使い果される可能性があることを警告するメッセージをalert_sid.logファイルに生成します。このパラメータには、Oracleプロセスに必要な優先順位レベルを割り当てておくことをお薦めします。

関連項目:

優先順位ポリシーと優先順位の範囲の詳細は、HP-UXのドキュメントと、rtsched(1)およびrtsched(2)のmanページを参照してください。

D.3 軽量タイマーの実装

Oracle Database 11gでは、動的初期化パラメータSTATISTICS_LEVELがTYPICAL(デフォルト)またはALLに設定されている場合、いつでもランタイム統計を収集できます。このパラメータの設定は、TIMED_STATISTICS初期化パラメータを暗黙的にtrueに設定します。HP-UXシステムのOracle Databaseでは、gethrtime( )システム・ライブラリ・コールを使用して、統計の収集中に経過時間を計算します。この軽量システム・ライブラリ・コールを使用すると、Oracleインスタンスを実行しながら、パフォーマンスに影響を与えることなく、いつでもランタイム統計を収集できます。

TIMED_STATISTICS初期化パラメータが明示的にtrueに設定されているときに、gethrtime( )システム・ライブラリ・コールを使用すると、使用していない場合に比べてOracleのパフォーマンスを最大10%向上させることができます。また、ランタイム統計の収集にgethrtime( )システム・ライブラリ・コールを使用しても、Oracle Databaseのオンライン・トランザクション処理のパフォーマンスが低下することはありません。

D.4 非同期入出力

非同期入出力擬似ドライバをHP-UX上で使用すると、Oracle DatabaseがRAWディスク・パーティションに対する入出力を非同期方式で実行できるようになり、入出力のオーバーヘッドが軽減してスループットが向上します。非同期入出力擬似ドライバは、HP-UXのサーバーとワークステーションの両方で使用できます。

この項では、次の項目について説明します。

- MLOCK権限の付与
- 非同期入出力の実装
- 非同期入出力の検証
- SGAの非同期フラグ

D.4.1 MLOCK権限の付与

Oracle Databaseで非同期入出力操作を処理できるようにするには、OSDBAグループ(dba)にMLOCK権限が必要です。dbaグループにMLOCK権限を付与するには:

1. rootユーザーでログインします。
2. テキスト・エディタを使用して/etc/privgroupファイルを開くか、必要な場合は作成します。
3. OSDBAグループの名前で始まる次の行を追加または編集し、MLOCK権限を指定します。

ノート:

このファイルでは、特定のグループに対する権限の指定に 1 行のみを使用する必要があります。このファイルに dba グループに関する行がすでに含まれている場合は、その行に MLOCK 権限を追加してください。

```
dba RTPRIO RTSCHED MLOCK
```

4. ファイルを保存してテキスト・エディタを終了します。
5. 次のコマンドを入力し、OSDBAグループに権限を付与します。

```
# /usr/sbin/setprivgrp -f /etc/privgroup
```

6. 次のコマンドを入力し、権限が正しく設定されていることを確認します。

```
# /usr/bin/getprivgrp dba
```

D.4.2 非同期入出力の実装

HP-UXで非同期入出力を使用するには、データベース・ファイルの記憶域オプションとしてRAWパーティションを使用する自動ストレージ管理ディスク・グループを使用する必要があります。

関連項目:

HP-UXシステムでの自動ストレージ管理とRAW論理ボリュームの構成の詳細は、『Oracle Databaseインストレーション・ガイド』を参照してください

いずれかの記憶域オプションで非同期入出力を実装する前に、System Administrator Managementユーティリティを使用して、HP-UXカーネルに非同期ディスク・ドライバを構成する必要があります。

ノート:

Oracle Database 11g リリース 1 では、ファイル・システムに対して DISK_ASYNCH_IO パラメータを FALSE に設定する必要がありませんでした。しかし、Oracle Database 11g リリース 2 からは、データベース・ファイルの格納にファイル・システムを使用する場合、初期化パラメータ DISK_ASYNCH_IO を FALSE に設定する必要があります。デフォルトでは、DISK_ASYNCH_IO の値は TRUE です。

DISK_ASYNCH_IO パラメータを TRUE に設定する必要があるのは、データベース・ファイルの格納に RAW パーティションを使用する場合のみです。

System Administrator Managementユーティリティを使用して非同期ディスク・ドライバを追加し、カーネルを構成するには:

1. rootユーザーで次のコマンドを実行します。

```
# sam
```

2. 「Kernel Configuration」領域を選択します。
3. 「Drivers」領域を選択します。
4. 非同期ディスク・ドライバ(asyncdsk)を選択します。

5. 「Actions」→「Add Driver to Kernel」の順に選択します。

6. 「List」→「Configurable Parameters」の順に選択します。

7. MAX_ASYNC_PORTSパラメータを選択します。

8. 「Action」→「Modify Configurable Parameter」の順に選択します。

9. 次のガイドラインに従ってパラメータに新しい値を指定し、「OK」をクリックします。

   MAX_ASYNC_PORTSパラメータは、構成可能なHP-UXカーネル・パラメータで、/dev/asyncファイルを同時に開くことができる最大プロセス数を制御します。

   最大数のプロセスが/dev/asyncファイルを開いた後で別のプロセスがそのファイルを開こうとすると、エラー・メッセージが表示されます。多数のシャドウ・プロセスやパラレル問合せスレーブが非同期入出力を実行しているシステムでは、このエラーが発生するとパフォーマンスが低下することがあります。このエラーは記録されません。このエラーを回避するには、/dev/asyncファイルにアクセスする可能性がある最大プロセス数を予測し、MAX_ASYNC_PORTSパラメータにその値を設定します。

10. 「Actions」→「Process a New Kernel」の順に選択します。

11. 次のいずれかのオプションを選択し、「OK」をクリックします。

    - Move Kernel Into Place and Shutdown System/Reboot Now
    - Do Not Move Kernel Into Place: Do Not Shutdown/Reboot Now

    2番目のオプションを選択すると、新しいカーネルvmunix_testとその作成に使用されるsystem.SAM構成ファイルの両方が、/stand/buildディレクトリに作成されます。

HP-UX非同期デバイス・ドライバを使用して非同期入出力操作をできるようにするには:

1. rootユーザーでログインします。
2. /dev/asyncが存在しない場合は、次のコマンドを使用して作成します。

```
# /sbin/mknod /dev/async c 101 0x0
```

デフォルトでは、マイナー番号は0(ゼロ)に設定されます次の表に、デバイス・ファイルの作成に使用できる様々な8ビットのマイナー番号を示します。

マイナー番号	説明
0x0	これは、/dev/async の HP-UX のデフォルト値です。
0x4	永久的に再試行するかわりに、ディスク・デバイス・タイムアウトが完了してエラー・コードが表示されるようになります。アプリケーション・レベルのディスク・ミラー化には、障害が発生したディスク・デバイスの修復をアプリケーションが永久的に待機する状況を回避するために、この設定が必要です。Oracle RDBMS ユーザーは、自動ストレージ管理のミラーリング/レプリケーション(内部冗長性)を使用する場合、この機能を有効にする必要があります。SGA はメモリー内でロックされます。 タイムアウト値は asyncdsk_io_timeout を使用してチューニングできます。これは、HP-UX 11i v3 で使用可能な、プライベートかつ動的なチューナブルです。

マイナー番号	説明
	asyncdsk_io_timeout のデフォルト値は 30 秒です。このチューナブルの有効値の範囲は 10 から 300 までです。kctune(1M)を起動すると asyncdsk_io_timeout チューナブル値を変更できます。 HP-UX 論理ボリューム・マネージャ(LVM)で構成されているボリュームの場合、タイムアウトは HP-UX LVM によって制御されるため、このフラグは必要ありません。
0x100	asyncdsk_open(2)のコール時の、非同期ドライバによるメモリー・ページのオンデマンド・ロックが可能になります。オーバーヘッドの低いルーチンが、I/O 操作の際にページをメモリー内にロックするために使用されます。 オンデマンド・ロックは、Oracle の自動メモリー管理機能(メモリー使用量の制御に init.ora ファイルの MEMORY_TARGET を使用)を使用する場合、非常に重要です。動的 nPar または動的 vPar 機能を利用する RDBMS デプロイメントでは、オンデマンド・ロックも構成する必要があります。 従来の RDBMS デプロイメントでは、オンデマンド・ロックの明らかな効果を踏まえてその使用を検討できます。一般に、SGA 全体がメモリー内に即時にロックされないため、RDBMS の起動が速くなります。ただし、メモリー・ページが I/O リクエストごとに動的にロック/ロック解除されるため、オンデマンド・ロックによってランタイム・パフォーマンスがわずかに低下するインスタンスもあります。
0x104	これは、0x100 および 0x4 の組合せです。どちらの機能も有効になります。

3. 次のコマンドを入力し、メジャー番号が101の/dev/asyncデバイス・ファイルが存在することを確認します。

```
# ls -l /dev/async
```

このコマンドの出力は、次のようになります。

```
crw------- 1 oracle   dba     101 0x000000 Oct 28 10:32  /dev/async
```

4. 必要に応じて、このデバイス・ファイルに対し、Oracleソフトウェア所有者およびOSDBAグループとの整合性のあるオペレーティング・システム所有者とアクセス権を指定します。

Oracleソフトウェア所有者がoracleで、OSDBAグループがdbaの場合は、次のコマンドを実行します。

```
# /usr/bin/chown oracle:dba /dev/async
# /usr/bin/chmod 660 /dev/async
```

D.4.3 非同期入出力の検証

非同期入出力を検証するには、最初に、HP-UX非同期ドライバがOracle Database用に構成されているかを検証します。次に、Oracle DatabaseがHP-UXデバイス・ドライバを介して非同期入出力を実行しているかを検証します。

- HP-UX非同期ドライバがOracle Database用に構成されているかの検証
- Oracle Databaseが非同期入出力を使用しているかの検証

D.4.3.1 HP-UX非同期ドライバがOracle Database用に構成されているかの検証

HP-UX非同期ドライバがOracle Database用に正しく構成されているかを検証するには:

1. パラレル問合せスレーブ・プロセスの数が少ないOracle Databaseを起動します。
2. 次のコマンドを入力して、GlancePlus/UXユーティリティを起動します。

```
$ gpm
```

3. メイン・ウィンドウで、「Reports」→「Process List」の順にクリックします。
4. 「Process List」ウィンドウで、パラレル問合せスレーブ・プロセスを1つ選択し、「Reports」→「Process Open Files」の順に選択します。

   パラレル問合せスレーブ・プロセスによって現在開かれているファイルのリストが表示されます。

5. オープン・ファイルのリストで、/dev/asyncファイルまたは101 0x104000モードをチェックします。

   いずれかがリストに含まれている場合は、パラレル問合せスレーブ・プロセスによって/dev/asyncファイルが開かれており、HP-UX非同期デバイス・ドライバは、Oracleプロセスが非同期入出力を実行できるように正しく構成されています。/dev/asyncファイルのファイル記述子番号をノートにとっておきます。

D.4.3.2 Oracle Databaseが非同期入出力を使用しているかの検証

Oracle DatabaseがHP-UX非同期デバイス・ドライバを介して非同期入出力を使用しているかを検証するには:

1. HP-UX tuscユーティリティを、前述の手順のGlancePlusで選択したものと同じOracleパラレル問合せスレーブにアタッチします。
2. 使用している環境で入出力バウンド問合せを実行します。
3. tusc出力でread/writeコールのパターンをチェックします。

   そのためには、たとえば、次のコマンドを入力します。pidは、非同期入出力を処理する予定のパラレル問合せスレーブのプロセスIDです。

```
$ tusc -p pid > tusc.output
```

4. 問合せの実行後、[Ctrl]+[C]を押してプロセスから切断し、tusc.outputファイルを開きます。

   次に、tusc.outputファイルのサンプルを示します。

```
( Attached to process 2052: "ora_p000_tpch" [ 64-bit ])
...................
.........................
[2052] read(9, "80¥0¥001¥013  ¥b¥0¥0¥0¥0¥0¥0¥0¥0".., 388) .. = 28
[2052] write(9, "¥0¥0¥00e¥0¥0¥0¥080¥0¥001¥013Ð ¥0".., 48) .. = 48
[2052] read(9, "80¥0¥001¥013¢ 18¥0¥0¥0¥0¥0¥0¥0¥0".., 388) .. = 28
[2052] write(9, "¥0¥0¥00e¥0¥0¥0¥080¥0¥001¥01bd4¥0".., 48) .. = 48
```

   DISK_ASYNCH_IO初期化パラメータが明示的にfalseに設定されていない場合(デフォルトのtrueに設定されている場合)、tusc.outputファイルには、同じファイル記述子(この例では9)の非同期read/writeコールのパターンが連続して表示されます。

   tusc.outputファイルのファイル記述子番号をGlancePlusの/dev/asyncファイルで使用されている番号にマップします。これらの番号は、特定のパラレル問合せスレーブ・プロセスについて一致している必要があります。これにより、HP-UX非同期ドライブを介した入出力が非同期であることが検証されます。同期入出力の場合またはDISK_ASYNCH_IO初期化パラメータが明示的にFALSEに設定されている場合、前述の非同期read/writeパターンは表示されません。かわりに、lseekまたはpread/pwriteのコールが表示されます。また、1つのファイル記述

子のみでなく、多数の異なるファイル記述子(read/writeの最初の引数)が表示されます。

D.4.4 SGAの非同期フラグ

HP-UX上のOracle Databaseでは、HP-UX非同期ドライバが提供する非ブロック・ポーリング機能を使用して、入出力操作のステータスがチェックされます。このポーリングは、送信された入出力操作のステータスに基づいて非同期ドライバが更新するフラグのチェックにより実行されます。HP-UXでは、このフラグが共有メモリーに読み込まれている必要があります。

Oracle Databaseでは、非同期フラグを各OracleプロセスのSGA内に構成します。HP-UX上のOracle Databaseは、真の非同期入出力メカニズムを備え、以前に発行された入出力操作の一部が完了していなくても、入出力リクエストを発行できます。これにより、パフォーマンスが向上し、パラレル入出力プロセスの優れたスケーラビリティが保証されます。

リリース8.1.7より前のOracle Databaseのリリースでは、入出力操作は、HP-UX非同期ドライバを使用して共有メモリーからのみ実行できました。Oracle Database 11gでは、新しいHP-UX非同期ドライバを使用して共有メモリーとプロセス専用領域の両方から入出力操作を実行できます。ただし、非同期ドライバを介した入出力操作は、実際は非同期ではありません。これは、Oracle Databaseでは、非同期ドライバに送信された入出力操作のステータスのチェックに、ブロック待機を実行する必要があるためです。その結果、データベース・ライター・プロセスなどの一部のOracleプロセスでは、実質的には同期入出力が処理されています。

D.5 大規模メモリーの割当てとOracle Databaseのチューニング

Oracle Database 11g以降で稼働するアプリケーションは、以前のリリースで稼働するアプリケーションと比べて非常に大きいメモリーを使用できます。これは、HP-UXシステム上のOracle Database 11gで、仮想メモリー・データ・ページのデフォルト設定がD (4KB)からL (4GB)に変更されたためです。

この項では、次の項目について説明します。

- デフォルトの大規模仮想メモリー・ページ・サイズ
- チューニングに関する推奨事項
- チューニング可能なベース・ページ・サイズ

D.5.1 デフォルトの大規模仮想メモリー・ページ・サイズ

デフォルトでは、Oracle Databaseは、プロセス専用メモリーの割当てに、HP-UXで使用できる最大の仮想メモリー・ページ・サイズ設定を使用します。これは、値L(最大)で定義されます。この値は、Oracle実行可能ファイルのリンク時に、LARGE_PAGE_FLAGSオプションの1つとして設定されます。

仮想メモリー・ページ・サイズがLに設定されていると、HP-UXは、使用可能なプロセス専用メモリーを、1GB制限または割り当てられたメモリー量の合計に達するまで、1MB、4MB、16MBなどのサイズのページに割り当てます。Oracle PGAに十分なメモリーが割り当てられていて、大きなデータ・ページ・サイズ単位でのメモリー割当てが可能な場合、オペレーティング・システムは一度に最大ページ・サイズを割り当てます。たとえば、Oracle PGAに48MBを割り当てている場合、システムでは、16MBを3ページ設定するか、より小さい倍数の単位サイズのページを組み合せることができます。たとえば、1MBを4ページ、4MBを3ページ、16MBを2ページにするなどです。PGAに64MBを割り当てた場合、データ・ページ単位サイズと使用可能なメモリーが一致するため、1ページ(64MB)がオペレーティング・システムによって割り当てられます。

一般に、大規模メモリー・ページによってアプリケーションのパフォーマンスは向上します。これは、オペレーティング・システムが処理する必要のある仮想メモリー変換時のエラー数が減り、より多くのCPUリソースをアプリケーションに解放できるためです。また、プロセス専用メモリーの割当てに必要なデータ・ページ数の合計も減ります。これにより、プロセッサ・レベルでのTranslation

Lookaside Bufferミスの可能性が減ります。

ただし、アプリケーションにメモリーの制約があり、非常に多くのプロセスを実行する傾向がある場合は、この大幅なページ・サイズの増加によってプロセスは大規模なメモリーの割当てを指示するため、メモリー不足エラー・メッセージが発生する可能性があります。このエラーが発生する場合は、ページ・サイズの値を小さくして、D(デフォルト)サイズの4KBからL(最大)サイズの4GBまでの間に設定する必要があります。

最小ページ・サイズ設定(4KB)を使用すると、最大ページ・サイズ設定を使用した場合よりもCPU使用率が20%以上高くなります。最大設定のLを使用すると、4MBの設定を使用した場合よりもメモリー使用率が50%以上高くなります。システムにメモリー制約がある場合は、使用できるメモリー・リソースの制約内で、ページ・サイズを特定のアプリケーションの要件と一致するように設定することをお薦めします。

たとえば、設定値Lでは問題が発生するアプリケーションで、4MBの仮想メモリー・ページ設定を使用すると、適切なパフォーマンスが得られる場合があります。

D.5.2 チューニングに関する推奨事項

永続的な専用SQL領域および大規模仮想メモリー・ページ・サイズへのメモリー割当ての増加に対応してチューニングを行うには、必要に応じて、Oracle Databaseの仮想メモリー・データ・ページ・サイズを減らすことをお薦めします。次のコマンドを使用して、ページ・サイズ設定を変更します。

```
# /usr/bin/chatr +pd newsize $ORACLE_HOME/bin/oracle
```

この例のnewsizeは、仮想メモリー・ページ・サイズの新しい値を表します。

chatrコマンドを次のように使用して、新しい設定を表示します。

```
# /usr/bin/chatr $ORACLE_HOME/bin/oracle
```

D.5.3 チューニング可能なベース・ページ・サイズ

ベース・ページ・サイズを大きくすると、メモリーを効率的に管理できます。base_pagesizeのデフォルト値は4KBです。HP-UX 11.31に導入された新機能により、kctune(1M)を起動してチューニング可能なbase_pagesizeを変更した後にコンピュータを再起動すると、ベース・ページのサイズを調整できます。

D.6 CPU_COUNT初期化パラメータおよびHP-UX動的プロセッサ再構成

HP-UX 11iでは、プロセッサ・セットの動的ランタイム再構成と、有効なユーザーによるプロセッサ・セット間のワークロードの動的再割当てをサポートしています。

HP-UX Virtual Partitionsを使用すると、ユーザーは、各自のシステムを複数の論理パーティションで構成し、各パーティションに独自のプロセッサ、メモリーおよび入出力リソースのセットを割り当てて、HP-UXオペレーティング・システムの個別のインスタンスを実行できます。vParsに組み込まれているHP-UXプロセッサ・セットを使用すると、仮想パーティションを再起動せずに、仮想パーティション間でプロセッサを動的に移行できます。これにより、アプリケーション間でリソースのパーティション化を効率的に行うことができるため、HP-UXサーバー上で稼働する各アプリケーションに必要なリソース割当てに対する干渉および保証を最小限に抑えることができます。

関連項目:

動的リソース・プロビジョニングの詳細は、『Oracle Database概要』を参照してください。

D.7 ネットワーク情報サービスの外部ネーミングのサポート

HP-UXでは、ネットワーク情報サービスの外部ネーミング・アダプタがサポートされています。

関連項目:

ネットワーク情報サービスの外部ネーミングの詳細は、『Oracle Database Net Services管理者ガイド』を参照してください

D.8 拡張ホスト名および拡張ノード名のアクティブ化と設定

デフォルトでは、システムのノード名とホスト名の長さの制限は8バイトと64バイトです。システム管理者は、システムを構成してどちらの制限も255バイトに拡張できます。

長い名前を設定できるようにするには、動的カーネル調整可能パラメータexpanded_node_host_namesを有効にする必要があります。

カーネル・パラメータを有効にするには、次のコマンドを実行します。

```
kctune expanded_node_host_names=1
```

カーネル・パラメータを無効にするには、次のコマンドを実行します。

```
kctune expanded_node_host_names=0
```

E Oracle ODBC Driverの使用

この付録では、Oracle ODBC Driverの使用方法について説明します。

内容は次のとおりです。

関連項目:

ODBCドライバの動作保証情報は、各プラットフォームの『Oracle Databaseインストレーション・ガイド』を参照してください。

E.1 サポートされていないOracle ODBCの機能

Oracle ODBC Driverは次のOracle ODBC 3.0機能をサポートしていません。

- 間隔データ型
- SQL_C_UBIGINTおよびSQL_C_SBIGINT Cデータ型識別子
- 共有接続
- 共有環境
- SQLSetConnectAttrのSQL_LOGIN_TIMEOUT属性
- 期限切れパスワード・オプション

Oracle ODBC Driverは、次の表に示すSQL関数をサポートしていません。

文字列関数	数値関数	時間関数、日付関数および間隔関数

文字列関数	数値関数	時間関数、日付関数および間隔関数
BIT_LENGTH	ACOS	CURRENT_DATE
CHAR_LENGTH	ASIN	CURRENT_TIME
CHARACTER_LENGTH	ATAN	CURRENT_TIMESTAMP
DIFFERENCE	ATAN2	EXTRACT
OCTET_LENGTH	COT	TIMESTAMPDIFF
POSITION	DEGREES	
	RADIANS	
	RAND	
	ROUND	

E.2 データ型の実装

この項では、DATE、TIMESTAMPおよび浮動小数点の各データ型について説明します。

DATEおよびTIMESTAMP

OracleのDATEおよびTIMESTAMPデータ型のセマンティクスは、同名のODBCデータ型と必ずしも正確に対応していません。OracleのDATEデータ型には、日付と時間の両方の情報が格納されています。これに対して、SQL_DATEデータ型に格納されているのは、日付情報のみです。OracleのTIMESTAMPデータ型にも日付と時間の情報が格納されていますが、その小数秒の精度は他方に比較して高くなります。Oracle ODBC Driverは、情報が失われるのを防ぐために、OracleのDATE列とTIMESTAMP列の両方のデータ型をSQL_TIMESTAMPとして報告します。同様に、Oracle ODBC DriverはSQL_TIMESTAMPパラメータをOracleのTIMESTAMP値としてバインドします。

関連項目:

パフォーマンスとチューニングに関するDATEおよびTIMESTAMPデータ型の詳細は、DATEおよびTIMESTAMPデータ型を参照してください

浮動小数点データ型

Oracle Database 12cリリース2 (12.2)以上に接続すると、Oracle ODBC Driverは、Oracleの浮動小数点データ型BINARY_FLOATとBINARY_DOUBLEをODBCデータ型SQL_REALとSQL_DOUBLEにそれぞれマップします。これより前のリリースでは、SQL_REALとSQL_DOUBLEがOracleの汎用数値データ型にマップされていました。

E.3 データ型に関する制限事項

Oracle ODBC DriverおよびOracle Databaseには、データ型に関する制限事項があります。次の表に制限事項を示します。

制限されるデータ型	説明
リテラル	Oracle Database では、SQL 文のリテラルは 4000 バイトに制限されています。
SQL_LONGVARCHAR および SQL_WLONGVARCHAR	Oracle では、列型が LONG の SQL_LONGVARCHAR データは 2,147,483,647 バイトに制限されます。また、Oracle では、列型が CLOB の SQL_LONGVARCHAR データは 4GB に制限されます。これは、クライアント・ワークステーションのメモリーによる制限です。
SQL_LONGVARCHAR および SQL_LONGVARBINARY	Oracle Database で使用できるのは、表ごとに 1 列の LONG データ列のみです。LONG データ型とは、SQL_LONGVARCHAR(LONG)および SQL_LONGVARBINARY(LONG RAW)です。かわりに、CLOB および BLOB 列の使用をお薦めします。1 つの表で使用できる CLOB および BLOB の列数に制限はありません。

E.4 SQLDriverConnect関数の接続文字列の書式

SQLDriverConnect関数は、Oracle ODBC Driverによって実装される関数の1つです。次の表に、SQLDriverConnect関数コールの接続文字列の引数に含めることができるキーワードを示します。

キーワード	意味	値
DSN	ODBC データソース名	ユーザー指定名 これは必須のキーワードです。
DBQ	TNS サービス名	ユーザー指定名 これは必須のキーワードです。
UID	ユーザーID またはユーザー名	ユーザー指定名 これは必須のキーワードです。
PWD	パスワード	ユーザー指定名 パスワードがない場合は PWD=;と指定します。 これは必須のキーワードです。
DBA	データベース属性	W は書込みアクセスを意味します。 R は読取り専用アクセスを意味します。

キーワード	意味	値
APA	アプリケーション属性	T はスレッド・セーフティを有効にすることを意味します。 F はスレッド・セーフティを無効にすることを意味します。
RST	結果セット	T は結果セットを有効にすることを意味します。 F は結果セットを無効にすることを意味します。
QTO	問合せタイムアウト・オプション	T は問合せタイムアウトを有効にすることを意味します。 F は問合せタイムアウトを無効にすることを意味します。
CSR	カーソルのクローズ	T はカーソルのクローズを有効にすることを意味します。 F はカーソルのクローズを無効にすることを意味します。
BAM	バッチ自動コミット・モード	`IfAllSuccessful` は、すべての文が成功した場合のみコミットすることを意味します(古い動作)。 `UpToFirstFailure` は、最初に失敗した文までコミットすることを意味します。これは ODBC バージョン 7 の動作です。 `AllSuccessful` は、成功したすべての文をコミットすることを意味します。
FBS	フェッチ・バッファ・サイズ	ユーザー指定の数値(0 バイト以上の値を指定します)。デフォルトは 60,000 バイトです。
FEN	フェイルオーバー	T はフェイルオーバーを有効にすることを意味します。 F はフェイルオーバーを無効にすることを意味します。
FRC	フェイルオーバー再試行数	ユーザー指定の数値。 デフォルトは 10 です。
FDL	フェイルオーバーの遅延。	ユーザー指定の数値。 デフォルトは 10 です。
LOB	LOB 書込み	T は LOB を有効にすることを意味します。

キーワード	意味	値
		F は LOB を無効にすることを意味します。
FWC	SQL_WCHAR 強制サポート	T は Force SQL_WCHAR を有効にすることを意味します。 F は Force SQL_WCHAR を無効にすることを意味します。
EXC	EXEC 構文	T は EXEC 構文を有効にすることを意味します。 F は EXEC 構文を無効にすることを意味します。
XSM	スキーマ・フィールド	Default はデフォルト値が使用されることを意味します。 Database はデータベース名が使用されることを意味します。 Owner は所有者名が使用されることを意味します。
MDI	METADATA ID デフォルトの設定	T は SQL_ATTR_METADATA_ID のデフォルト値が SQL_TRUE であることを意味します。 F は SQL_ATTR_METADATA_ID のデフォルト値が SQL_FALSE であることを意味します。
DPM	SQLDescribeParam の無効化	T は SQLDescribeParam を無効にすることを意味します。 F は SQLDescribeParam を有効にすることを意味します。
BTD	TIMESTAMP を DATE としてバインド	T は SQL_TIMESTAMP が Oracle の DATE としてバインドされることを意味します。 F は SQL_TIMESTAMP が Oracle の TIMESTAMP としてバインドされることを意味します。
NUM	数値の設定	NLS は、グローバリゼーション・サポートの数値の設定が使用されることを意味します(小数点およびグループのセパレータを判断するため)。

E.5 プログラムでのロック・タイムアウトの削減

Oracle Databaseは、トランザクション間のロックの競合が解決されるまで無期限に待機します。ただし、Oracle Databaseがロックの解決を待機する時間は制限できます。制限するには、ODBCのSQLSetStmtAttr関数のコール時にSQL_ATTR_QUERY_TIMEOUT属性を設定してから、データソースに接続します。

E.6 ODBCアプリケーションのリンク

プログラムをリンクする場合は、そのプログラムをDriver Managerライブラリ`libodbc.so`にリンクする必要があります。

E.7 ROWIDに関する情報の取得

ODBCの`SQLSpecialColumns`関数は、表内の列に関する情報を返します。Oracle ODBC Driverで使用すると、この関数はOracle表に関連付けられたOracle ROWIDに関する情報を返します。

E.8 WHERE句のROWID

ROWIDは、SQL文の`WHERE`句で使用できます。ただし、ROWID値はパラメータ・マーカー内に存在している必要があります。

E.9 結果セットの有効化

Oracle参照カーソル(結果セットとも呼ばれます)によって、アプリケーションでは、ストアド・プロシージャとストアド・ファンクションを使用してデータを取得できます。ここでは、参照カーソルを使用してODBCを介して結果セットを有効にする方法を説明します。

- ストアド・プロシージャをコールするには、ODBC構文を使用する必要があります。システム固有のPL/SQLは、ODBCを介してサポートされていません。次のコード例は、プロシージャまたはファンクションを、パッケージなしでコールする方法、およびパッケージ内でコールする方法を示します。このコード例でのパッケージ名はRSETです。

```
Procedure call:
{CALL Example1(?)}
{CALL RSET.Example1(?)}
Function Call:
{? = CALL Example1(?)}
{? = CALL RSET.Example1(?)}
```

- PL/SQLの参照カーソル・パラメータは、プロシージャのコール時には省略されます。たとえば、プロシージャ`Example2`には4つのパラメータが定義されているとします。パラメータ1と3は参照カーソル・パラメータで、パラメータ2と4は文字列です。コールは、次のように指定されます。

```
{CALL RSET.Example2("Literal 1", "Literal 2")}
```

次のサンプル・アプリケーションは、Oracle ODBC Driverを使用して結果セットを返す方法を示します。

```
/*
* Sample Application using Oracle reference cursors through ODBC
*
* Assumptions:
*
* 1) Oracle Sample database is present with data loaded for the EMP table.
*
* 2) Two fields are referenced from the EMP table, ename and mgr.
*
* 3) A data source has been setup to access the sample database.
*
*
* Program Description:
*
* Abstract:
*
* This program demonstrates how to return result sets using
* Oracle stored procedures
*
* Details:
```

```
*
* This program:
* Creates an ODBC connection to the database.
* Creates a Packaged Procedure containing two result sets.
* Executes the procedure and retrieves the data from both result sets.
* Displays the data to the user.
* Deletes the package then logs the user out of the database.
*
*
* The following is the actual PL/SQL this code generates to
* create the stored procedures.
*
DROP PACKAGE   ODBCRefCur;
CREATE PACKAGE ODBCRefCur AS
     TYPE ename_cur IS REF CURSOR;
     TYPE mgr_cur   IS REF CURSOR;
PROCEDURE EmpCurs(Ename IN OUT ename_cur, Mgr IN OUT mgr_cur, pjob IN VARCHAR2);
END;
/
CREATE PACKAGE BODY ODBCRefCur AS
PROCEDURE EmpCurs(Ename IN OUT ename_cur, Mgr IN OUT mgr_cur, pjob IN VARCHAR2)
     AS
       BEGIN
          IF NOT Ename%ISOPEN
          THEN
              OPEN Ename for SELECT ename from emp;
          END IF;
          IF NOT Mgr%ISOPEN
          THEN
              OPEN Mgr for SELECT mgr from emp where job = pjob;
          END IF;
       END;
END;
/
*
*  End PL/SQL for Reference Cursor.
*/
/*
 * Include Files
 */
#include <stdio.h>
#include <sql.h>
#include <sqlext.h>
/*
 * Defines
 */
#define JOB_LEN 9
#define DATA_LEN 100
#define SQL_STMT_LEN 500
/*
 * Procedures
 */
void DisplayError( SWORD HandleType, SQLHANDLE hHandle, char *Module );
/*
 * Main Program
 */
int main()
{
SQLHENV hEnv;
SQLHDBC hDbc;
SQLHSTMT hStmt;
SQLRETURN rc;
char *DefUserName ="jones";
char *DefPassWord ="password";
SQLCHAR ServerName[DATA_LEN];
SQLCHAR *pServerName=ServerName;
SQLCHAR UserName[DATA_LEN];
```

```
SQLCHAR *pUserName=UserName;
SQLCHAR PassWord[DATA_LEN];
SQLCHAR *pPassWord=PassWord;
char Data[DATA_LEN];
SQLINTEGER DataLen;
char error[DATA_LEN];
char *charptr;
SQLCHAR SqlStmt[SQL_STMT_LEN];
SQLCHAR *pSqlStmt=SqlStmt;
char *pSalesMan = "SALESMAN";
SQLINTEGER sqlnts=SQL_NTS;
/*
 * Allocate the Environment Handle
 */
rc = SQLAllocHandle( SQL_HANDLE_ENV, SQL_NULL_HANDLE, &hEnv );
if (rc != SQL_SUCCESS)
{
  printf( "Cannot Allocate Environment Handle¥n");
  printf( "¥nHit Return to Exit¥n");
  charptr = gets ((char *)error);
  exit(1);
}
/*
 * Set the ODBC Version
 */
rc = SQLSetEnvAttr( hEnv,SQL_ATTR_ODBC_VERSION,(void *)SQL_OV_ODBC3,0);
if (rc != SQL_SUCCESS)
{
  printf( "Cannot Set ODBC Version¥n");
  printf( "¥nHit Return to Exit¥n");
  charptr = gets ((char *)error);
  exit(1);
}
/*
 * Allocate the Connection handle
 */
rc = SQLAllocHandle( SQL_HANDLE_DBC, hEnv, &hDbc );
if (rc != SQL_SUCCESS)
{
  printf( "Cannot Allocate Connection Handle¥n");
  printf( "¥nHit Return to Exit¥n");
  charptr = gets ((char *)error);
  exit(1);
}
/*
 * Get User Information
 */
strcpy ((char *) pUserName, DefUserName );
strcpy ((char *) pPassWord, DefPassWord );
/*
 * Data Source name
 */
printf( "¥nEnter the ODBC Data Source Name¥n" );
charptr = gets ((char *) ServerName);
/*
 * User Name
 */
printf ( "¥nEnter User Name Default [%s]¥n", pUserName);
charptr = gets ((char *) UserName);
if (*charptr == '¥0')
{
  strcpy ((char *) pUserName, (char *) DefUserName );
}
/*
 * Password
 */
printf ( "¥nEnter Password Default [%s]¥n", pPassWord);
```

```
charptr = gets ((char *)PassWord);
if (*charptr == '¥0')
{
  strcpy ((char *) pPassWord, (char *) DefPassWord );
}
/*
 * Connection to the database
 */
rc = SQLConnect( hDbc,pServerName,(SQLSMALLINT) strlen((char
*)pServerName),pUserName,(SQLSMALLINT) strlen((char
*)pUserName),pPassWord,(SQLSMALLINT) strlen((char *)pPassWord));
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_DBC, hDbc, "SQLConnect");
}
/*
 * Allocate a Statement
 */
rc = SQLAllocHandle( SQL_HANDLE_STMT, hDbc, &hStmt );
if (rc != SQL_SUCCESS)
{
  printf( "Cannot Allocate Statement Handle¥n");
  printf( "¥nHit Return to Exit¥n");
  charptr = gets ((char *)error);
  exit(1);
}
/*
 * Drop the Package
 */
strcpy( (char *) pSqlStmt, "DROP PACKAGE ODBCRefCur");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
/*
 * Create the Package Header
 */
strcpy( (char *) pSqlStmt, "CREATE PACKAGE ODBCRefCur AS¥n");
strcat( (char *) pSqlStmt, " TYPE ename_cur IS REF CURSOR;¥n");
strcat( (char *) pSqlStmt, " TYPE mgr_cur IS REF CURSOR;¥n¥n");
strcat( (char *) pSqlStmt, " PROCEDURE EmpCurs (Ename IN OUT ename_cur,");
strcat( (char *) pSqlStmt, "Mgr IN OUT mgr_cur,pjob IN VARCHAR2);¥n¥n");
strcat( (char *) pSqlStmt, "END;¥n");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLExecDirect");
}
/*
 * Create the Package Body
 */
strcpy( (char *) pSqlStmt, "CREATE PACKAGE BODY ODBCRefCur AS¥n");
strcat( (char *) pSqlStmt, " PROCEDURE EmpCurs (Ename IN OUT ename_cur,");
strcat( (char *) pSqlStmt, "Mgr IN OUT mgr_cur, pjob IN VARCHAR2)¥n AS¥n  BEGIN¥n");
strcat( (char *) pSqlStmt, " IF NOT Ename%ISOPEN¥n THEN¥n");
strcat( (char *) pSqlStmt, " OPEN Ename for SELECT ename from emp;¥n");
strcat( (char *) pSqlStmt, " END IF;¥n¥n");
strcat( (char *) pSqlStmt, " IF NOT Mgr%ISOPEN¥n THEN¥n");
strcat( (char *) pSqlStmt, " OPEN Mgr for SELECT mgr from emp where job = pjob;¥n");
strcat( (char *) pSqlStmt, " END IF;¥n");
strcat( (char *) pSqlStmt, " END;¥n");
strcat( (char *) pSqlStmt, "END;¥n");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLExecDirect");
}
/*
 * Bind the Parameter
 */
```

```
rc =
SQLBindParameter(hStmt,1,SQL_PARAM_INPUT,SQL_C_CHAR,SQL_CHAR,JOB_LEN,0,pSalesMan,0,&s
qlnts);
/*
 * Call the Store Procedure which executes the Result Sets
 */
strcpy( (char *) pSqlStmt, "{CALL ODBCRefCur.EmpCurs(?)}");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLExecDirect");
}
/*
 * Bind the Data
 */
rc = SQLBindCol( hStmt,1,SQL_C_CHAR,Data,sizeof(Data),&DataLen);
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLBindCol");
}
/*
 * Get the data for Result Set 1
 */
printf( "¥nEmployee Names¥n¥n");
while ( rc == SQL_SUCCESS )
{
  rc = SQLFetch( hStmt );
  if ( rc == SQL_SUCCESS )
  {
    printf("%s¥n", Data);
  }
  else
  {
    if (rc != SQL_NO_DATA)
    {
      DisplayError(SQL_HANDLE_STMT, hStmt, "SQLFetch");
    }
  }
}
printf( "¥nFirst Result Set - Hit Return to Continue¥n");
charptr = gets ((char *)error);
/*
 * Get the Next Result Set
 */
rc = SQLMoreResults( hStmt );
if (rc != SQL_SUCCESS)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLMoreResults");
}
/*
 * Get the data for Result Set 2
 */
printf( "¥nManagers¥n¥n");
while ( rc == SQL_SUCCESS )
{
  rc = SQLFetch( hStmt );
  if ( rc == SQL_SUCCESS )
  {
    printf("%s¥n", Data);
  }
  else
  {
    if (rc != SQL_NO_DATA)
    {
    DisplayError(SQL_HANDLE_STMT, hStmt, "SQLFetch");
    }
  }
```

```
}
printf( "¥nSecond Result Set - Hit Return to Continue¥n");
charptr = gets ((char *)error);
/*
 * Should Be No More Results Sets
 */
rc = SQLMoreResults( hStmt );
if (rc != SQL_NO_DATA)
{
  DisplayError(SQL_HANDLE_STMT, hStmt, "SQLMoreResults");
}
/*
 * Drop the Package
 */
strcpy( (char *) pSqlStmt, "DROP PACKAGE ODBCRefCur");
rc = SQLExecDirect(hStmt, pSqlStmt, strlen((char *)pSqlStmt));
/*
 * Free handles close connections to the database
 */
SQLFreeHandle( SQL_HANDLE_STMT, hStmt );
SQLDisconnect( hDbc );
SQLFreeHandle( SQL_HANDLE_DBC, hDbc );
SQLFreeHandle( SQL_HANDLE_ENV, hEnv );
printf( "¥nAll Done - Hit Return to Exit¥n");
charptr = gets ((char *)error);
return(0);
}
/*
 * Display Error Messages
 */
void DisplayError( SWORD HandleType, SQLHANDLE hHandle, char *Module )
{
SQLCHAR MessageText[255];
SQLCHAR SQLState[80];
SQLRETURN rc=SQL_SUCCESS;
long NativeError;
SWORD RetLen;
SQLCHAR error[25];
char *charptr;
rc =
SQLGetDiagRec(HandleType,hHandle,1,SQLState,&NativeError,MessageText,255,&RetLen);
printf( "Failure Calling %s¥n", Module );
if (rc == SQL_SUCCESS || rc == SQL_SUCCESS_WITH_INFO)
{
  printf( "¥t¥t¥t State: %s¥n", SQLState);
  printf( "¥t¥t¥t Native Error: %d¥n", NativeError );
  printf( "¥t¥t¥t Error Message: %s¥n", MessageText );
}
printf( "¥nHit Return to Exit¥n");
charptr = gets ((char *)error);
exit(1);
}
```

E.10 EXEC構文の有効化

使用しているSQL ServerのEXEC文の構文を変更せずに同等のOracleプロシージャ・コールに容易に変換できる場合、その構文は、このオプションを有効にすると、Oracle ODBC Driverで変換できます。

SQL Serverプロシージャの完全な名前は、次に示す最大4つの識別子で構成されます。

- サーバー名
- データベース名

- 所有者名
- プロシージャ名

この名前の書式は次のとおりです。

```
[[[server.][database].][owner_name].]procedure_name
```

Microsoft SQL ServerデータベースからOracle Databaseに移行する際、各SQL Serverプロシージャまたはファンクションの定義は同等のOracle Database構文に変換され、Oracle Databaseのスキーマで定義されます。移行したプロシージャは、多くの場合、次のいずれかの方法で再編成(およびスキーマで作成)されます。

- すべてのプロシージャが1つのスキーマに移行されます(デフォルト・オプション)。
- 1つのSQL Serverデータベースに定義されているすべてのプロシージャが、そのデータベース名の付いたスキーマに移行されます。
- 特定のユーザーが所有するすべてのプロシージャが、そのユーザー名の付いたスキーマに移行されます。

移行したプロシージャを編成するこれら3通りの方法をサポートするためには、いずれかのスキーマ名オプションを指定してプロシージャ名を変換できます。変換したOracleプロシージャ・コールのオブジェクト名では、大/小文字が区別されません。

E.11 サポートされている機能

この項では、Oracle ODBC Driverでサポートされている機能について説明します。内容は次のとおりです。

- APIへの準拠
- ODBC API関数の実装
- ODBC SQL構文の実装
- データ型の実装

E.11.1 APIへの準拠

Oracle ODBC Driverリリース10.2.0.1.0以上では、コア、レベル2およびレベル1のすべての関数をサポートしています。

E.11.2 ODBC API関数の実装

次の表に、Oracle ODBC Driverが特定の関数を実装する方法を示します。

機能	説明
SQLConnect	SQLConnect で必要なのは、DBQ、ユーザーID およびパスワードのみです。
SQLDriverConnect	SQLDriverConnect では、DSN、DBQ、UID および PWD の各キーワードが使用されます。
SQLSpecialColumns	SQL_BEST_ROWID 属性を指定して SQLSpecialColumns をコールすると、常に ROWID 列を返します。

機能	説明
SQLProcedures および SQLProcedureColumns	次の項の説明を参照してください。
すべてのカタログ・ファンクション	SQL_ATTR_METADATA_ID 文の属性が SQL_TRUE に設定されている場合、文字列の引数は、識別子の引数として処理されますが、このケースはあまり重要ではありません。この場合、アンダースコア(_)およびパーセント記号(%)は検索用パターン文字ではなく、実際の文字として処理されます。これに対して、この属性を SQL_FALSE に設定すると、文字列の引数は、通常の引数またはパターン値の引数となり、リテラルに処理されるため、このケースは重要です。

SQLProceduresおよびSQLProcedureColumns

SQLProceduresコールおよびSQLProcedureColumnsコールは、パッケージ内に含まれている場合でも、すべてのプロシージャおよびファンクションに関する情報を検索して返すように変更されました。これより前のリリースでは、これらのコールで検索されるのは、パッケージ外のプロシージャとファンクションのみでした。次の例では、SQL_ATTR_METADATA_ID属性をSQL_FALSEに設定した場合に返されるプロシージャまたはファンクションを示します。

この例では、次のストアド・プロシージャがあるとします。

```
"BAR"
"BARX"
"XBAR"
"XBARX"
"SQLPROCTEST.BAR"
"SQLPROCTEST.BARX"
"SQLPROCTEST.XBAR"
"SQLPROCTEST.XBARX"
```

%または%%%%%と指定して検索すると、これら8つのプロシージャがすべて返されます。

%_または_%と指定して検索すると、次のプロシージャが返されます。

```
BAR
BARX
XBAR
XBARX
```

.、.%、%.%、SQLPROC%.またはSQLPROC%.%と指定して検索すると、次のプロシージャが返されます。

```
SQLPROCTEST.BAR
SQLPROCTEST.BARX
SQLPROCTEST.XBAR
SQLPROCTEST.XBARX
```

%BARと指定して検索すると、次のプロシージャが返されます。

```
BAR
XBAR
```

.%BARまたは%.%BARと指定して検索すると、次のプロシージャが返されます。

```
SQLPROCTEST.BAR
SQLPROCTEST.XBAR
```

SQLPROC%または.SQLPROC%と指定して検索すると、次のプロシージャが返されます。

```
nothing (0 rows)
```

E.11.3 ODBC SQL構文の実装

比較述語で比較の2番目の式としてパラメータ・マーカーが使用され、そのパラメータの値がSQLBindParameterを使用してSQL_NULL_DATAに設定されている場合、比較は失敗します。これは、ODBC SQLのNULL述語構文と一致しています。

E.11.4 データ型の実装

プログラマがデータ型を実装する際に最も考慮する必要があるのは、CHAR、VARCHARおよびVARCHAR2の各データ型です。

SQL_VARCHARの値がfSqlTypeの場合、SQLGetTypeInfoはOracle Databaseデータ型VARCHAR2を返します。SQL_CHARの値がfSqlTypeの場合、SQLGetTypeInfoはOracle Databaseデータ型CHARを返します。

E.12 Unicodeのサポート

この項では、Unicodeのサポートについて説明します。次の項目が含まれます。

- ODBC環境内でのUnicodeのサポート
- ODBC APIでのUnicodeのサポート
- SQLGetDataのパフォーマンス
- Unicodeのサンプル

E.12.1 ODBC環境内でのUnicodeのサポート

ODBC Driver Managerを使用すると、すべてのODBCドライバは、Unicodeをサポートしているかどうかに関係なくUnicode準拠と同様に動作します。これによって、ODBCアプリケーションは、基礎となるODBCドライバのUnicode機能に関係なく記述できます。

Driver ManagerがANSI ODBCドライバに対するUnicodeサポートをエミュレートできる範囲は、Unicodeデータとローカル・コード・ページ間で可能な変換内容によって制限されます。Driver Managerがデータをローカル・コード・ページに変換する際、データが失われる場合があります。基礎となるODBCドライバがUnicodeをサポートしていないかぎり、Unicodeの完全なサポートは不可能です。Oracle ODBC Driverは、Unicodeを完全にサポートしています。

E.12.2 ODBC APIでのUnicodeのサポート

ODBC APIは、WおよびA接尾辞の変換を使用して、UnicodeおよびANSIエントリ・ポイントの両方をサポートします。ODBCアプリケーション開発者は、接尾辞を指定してエントリ・ポイントを明示的にコールすることはありません。UNICODEおよび_UNICODEプリプロセッサ定義を使用してコンパイルされたODBCアプリケーションによって、適切なコールが生成されます。たとえば、SQLPrepareへのコールは、SQLPrepareWとしてコンパイルされます。

CデータSQL_C_WCHARがODBCインタフェースに追加されたため、アプリケーションでは、入力パラメータをUnicodeとしてエンコードするように指定するか、またはUnicodeとして返された列データを要求できます。マクロのSQL_C_TCHARは、UnicodeおよびANSIの両方で作成する必要があるアプリケーションで有効です。SQL_C_TCHARマクロは、Unicodeアプリケーションの場合はSQL_C_WCHARとして、ANSIアプリケーションの場合はSQL_C_CHARとしてコンパイルされます。

SQLデータSQL_WCHAR、SQL_WVARCHARおよびSQL_WLONGVARCHARがODBCインタフェースに追加され、表内で定義された列がUnicodeで表現されるようになりました。これらの値は、SQLDescribeCol、SQLColAttribute、

SQLColumnsおよびSQLProcedureColumnsへのコールから返すことも可能です。

Unicodeエンコーディングは、SQL列型のNCHAR、NVARCHAR2およびNCLOBに対してサポートされています。さらに、文字セマンティクスが列定義で指定されている場合、UnicodeエンコーディングはSQL列型のCHARおよびVARCHAR2に対してもサポートされています。

Oracle ODBC Driverは、これらのSQL列型をサポートしてODBC SQLデータ型にマップします。次の表に、サポートされているSQLデータ型および対応するODBC SQLデータ型を示します。

SQLデータ型	ODBC SQLデータ型
CHAR	SQL_CHAR または SQL_WCHAR
VARCHAR2	SQL_VARCHAR または SQL_WVARCHAR
NCHAR	SQL_WCHAR
NVARCHAR2	SQL_WVARCHAR
NCLOB	SQL_WLONGVARCHAR

E.12.3 SQLGetDataのパフォーマンス

SQLGetData関数を使用すると、ODBCアプリケーションはデータ型を指定して、データのフェッチ後に列を取得できます。OCIでは、Oracle ODBC Driverはデータ型を指定してからフェッチする必要があります。この場合、Oracle ODBC Driverは(データベースで定義された)列のデータ型に関する情報を使用し、OCIを介して列をフェッチする最適なデフォルト方法を判断します。

文字データを含む列がSQLBindColによってバインドされていない場合、Oracle ODBC Driverは、その列をUnicodeとしてフェッチするか、またはローカル・コード・ページとしてフェッチするかを判断する必要があります。ドライバは、列をUnicodeとして受け入れるように常にデフォルト設定できます。ただし、この設定では、2回の不要な変換が実行されます。たとえば、データベース内のデータがANSIとしてエンコードされた場合、データをOracle ODBC DriverにフェッチするためにANSIからUnicodeへの変換が実行されます。次に、ODBCアプリケーションがそのデータをSQL_C_CHARとして要求すると、データを元のエンコーディングに戻すための変換が実行されます。

Oracle Database Clientのデフォルト・エンコーディングは、データをフェッチする際に使用されます。しかし、ODBCアプリケーションは、WCHARデータ型として列またはパラメータをバインドすることにより、このデフォルトを上書きして、Unicodeとしてデータをフェッチする場合があります。

E.12.4 Unicodeのサンプル

Oracle ODBC Driver自体がTCHARマクロを使用して実装されているため、ODBCアプリケーション・プログラムでは、TCHARを使用して、このドライバを利用することをお薦めします。

次の例では、UNICODEおよび_UNICODEを指定してコンパイルするとWCHARデータ型になるTCHARの使用方法を示します。

例E-1 データベースへの接続

このコードを使用するには、SQLConnectにUnicodeリテラルのみ指定する必要があります。

```
HENV         envHnd;
HDBC         conHnd;
HSTMT        stmtHnd;
RETCODE      rc;
rc  = SQL_SUCCESS;
// ENV is allocated
rc = SQLAllocEnv(&envHnd);
// Connection Handle is allocated
rc = SQLAllocConnect(envHnd, &conHnd);
rc = SQLConnect(conHnd, _T("stpc19"), SQL_NTS, _T("jones"), SQL_NTS, _T("password"),
SQL_NTS);
.
.
.
if (conHnd)
  SQLFreeConnect(conHnd);
if (envHnd)
  SQLFreeEnv(envHnd);
```

例E-2 単純な取得

次の例では、EMP表から従業員名と役職名を取得します。すべてのODBC関数にTCHAR準拠のデータを指定する必要があることを除いて、ANSIの場合と違いはありません。Unicodeアプリケーションの場合は、SQLBindColのコール時にバッファ長をBYTE長に指定する必要があります。たとえば、sizeof(ename)の場合は次のとおりです。

```
/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLExecDirect
**   SQLBindCol
**   SQLFetch
**
*/
static SQLTCHAR *sqlStmt = _T("SELECT ename, job FROM emp");
SQLTCHAR   ename[50];
SQLTCHAR     job[50];
SQLINTEGER enamelen, joblen;

_tprintf(_T("Retrieve ENAME and JOB using SQLBindCol 1.../n[%s]/n"), sqlStmt);

// Step 1: Prepare and Execute
rc = SQLExecDirect(stmtHnd, sqlStmt, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 2: Bind Columns
rc = SQLBindCol(stmtHnd,
                1,
                SQL_C_TCHAR,
                ename,
                sizeof(ename),
                &enamelen);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLBindCol(stmtHnd,
                2,
                SQL_C_TCHAR,
                job,
                sizeof(job),
                &joblen);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

do
{
  // Step 3: Fetch Data
```

```
  rc = SQLFetch(stmtHnd);
  if (rc == SQL_NO_DATA)
    break;
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);
  _tprintf(_T("ENAME = %s, JOB = %s/n"), ename, job);
} while (1);
_tprintf(_T("Finished Retrieval/n/n"));
```

例E-3 SQLGetDataを使用した取得(フェッチ後のバインド)

この例では、SQLGetDataの使用方法を示します。Unicode固有の事項に関しては、ANSIアプリケーションとの違いはありません。

```
/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLExecDirect
**   SQLFetch
**   SQLGetData
*/
static SQLTCHAR *sqlStmt = _T("SELECT ename,job FROM emp"); // same as Case 1.
SQLTCHAR         ename[50];
SQLTCHAR         job[50];

_tprintf(_T("Retrieve ENAME and JOB using SQLGetData.../n[%s]/n"), sqlStmt);
if (rc != SQL_SUCCESS)
{
  _tprintf(_T("Failed to allocate STMT/n"));
  goto exit2;
}

// Step 1: Prepare and Execute
rc = SQLExecDirect(stmtHnd, sqlStmt, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

do
{

  // Step 2: Fetch
  rc = SQLFetch(stmtHnd);
  if (rc == SQL_NO_DATA)
    break;
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);

  // Step 3: GetData
  rc = SQLGetData(stmtHnd,
    1,
    SQL_C_TCHAR,
    (SQLPOINTER)ename,
    sizeof(ename),
    NULL);
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);
  rc = SQLGetData(stmtHnd,
    2,
    SQL_C_TCHAR,
    (SQLPOINTER)job,
    sizeof(job),
    NULL);
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);
  _tprintf(_T("ENAME = %s, JOB = %s/n"), ename, job);
} while (1);
_tprintf(_T("Finished Retrieval/n/n"));
```

例E-4 単純な更新

この例では、データの更新方法を示します。SQLBindParameterのデータ長は、UnicodeアプリケーションでもBYTE長で指定する必要があります。

```
/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLPrepare
**   SQLBindParameter
**   SQLExecute
*/
static SQLTCHAR *sqlStmt = _T("INSERT INTO emp(empno,ename,job) VALUES(?,?,?)");
static SQLTCHAR *empno   = _T("9876");      // Emp No
static SQLTCHAR *ename   = _T("ORACLE");    // Name
static SQLTCHAR *job     = _T("PRESIDENT"); // Job

_tprintf(_T("Insert User ORACLE using SQLBindParameter.../n[%s]/n"), sqlStmt);

// Step 1: Prepare
rc = SQLPrepare(stmtHnd, sqlStmt, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 2: Bind Parameter
rc = SQLBindParameter(stmtHnd,
                      1,
                      SQL_PARAM_INPUT,
                      SQL_C_TCHAR,
                      SQL_DECIMAL,
                      4,               // 4 digit
                      0,
                      (SQLPOINTER)empno,
                      0,
                      NULL);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLBindParameter(stmtHnd,
                      2,
                      SQL_PARAM_INPUT,
                      SQL_C_TCHAR,
                      SQL_CHAR,
                      lstrlen(ename)*sizeof(TCHAR),
                      0,
                      (SQLPOINTER)ename,
                      lstrlen(ename)*sizeof(TCHAR),
                      NULL);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLBindParameter(stmtHnd,
                      3,
                      SQL_PARAM_INPUT,
                      SQL_C_TCHAR,
                      SQL_CHAR,
                      lstrlen(job)*sizeof(TCHAR),
                      0,
                      (SQLPOINTER)job,
                      lstrlen(job)*sizeof(TCHAR),
                      NULL);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 3: Execute
rc = SQLExecute(stmtHnd);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
```

例E-5 長いデータ(CLOB)の更新と取得

ここでは、CLOBなどの長いデータをOracle Databaseで更新および取得する最も複雑な例を示します。データ長は常にBYTE長であるため、BYTE長を導出するには式lstrlen(TCHAR data)*sizeof(TCHAR)が必要です。

```
/*
** Execute SQL, bind columns, and Fetch.
** Procedure:
**
**   SQLPrepare
**   SQLBindParameter
**   SQLExecute
**   SQLParamData
**   SQLPutData
**
**   SQLExecDirect
**   SQLFetch
**   SQLGetData
*/
static SQLTCHAR *sqlStmt1  = _T("INSERT INTO clobtbl(clob1) VALUES(?)");
static SQLTCHAR *sqlStmt2  = _T("SELECT clob1 FROM clobtbl");
SQLTCHAR         clobdata[1001];
SQLTCHAR         resultdata[1001];
SQLINTEGER       ind = SQL_DATA_AT_EXEC;
SQLTCHAR        *bufp;
int              clobdatalen, chunksize, dtsize, retchklen;

_tprintf(_T("Insert CLOB1 using SQLPutData.../n[%s]/n"), sqlStmt1);

// Set CLOB Data
{
  int i;
  SQLTCHAR ch;
  for (i=0, ch=_T('A'); i< sizeof(clobdata)/sizeof(SQLTCHAR); ++i, ++ch)
  {
    if (ch > _T('Z'))
      ch = _T('A');
    clobdata[i] = ch;
  }
  clobdata[sizeof(clobdata)/sizeof(SQLTCHAR)-1] = _T('/0');
}
clobdatalen = lstrlen(clobdata);  // length of characters
chunksize   = clobdatalen / 7;    // 7 times to put
// Step 1: Prepare
rc = SQLPrepare(stmtHnd, sqlStmt1, SQL_NTS);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
// Step 2: Bind Parameter with SQL_DATA_AT_EXEC
rc = SQLBindParameter(stmtHnd,
                      1,
                      SQL_PARAM_INPUT,
                      SQL_C_TCHAR,
                      SQL_LONGVARCHAR,
                      clobdatalen*sizeof(TCHAR),
                      0,
                      (SQLPOINTER)clobdata,
                      clobdatalen*sizeof(TCHAR),
                      &ind);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
// Step 3: Execute
rc = SQLExecute(stmtHnd);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 4: ParamData (initiation)
rc = SQLParamData(stmtHnd, (SQLPOINTER*)&bufp); // set value
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

for (dtsize=0, bufp = clobdata;
     dtsize < clobdatalen;
```

```
    dtsize += chunksize, bufp += chunksize)
{
  int len;
  if (dtsize+chunksize<clobdatalen)
    len = chunksize;
  else
    len = clobdatalen-dtsize;

  // Step 5: PutData
  rc = SQLPutData(stmtHnd, (SQLPOINTER)bufp, len*sizeof(TCHAR));
  checkSQLErr(envHnd, conHnd, stmtHnd, rc);
}

// Step 6: ParamData (temination)
rc = SQLParamData(stmtHnd, (SQLPOINTER*)&bufp);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

rc = SQLFreeStmt(stmtHnd, SQL_CLOSE);
_tprintf(_T("Finished Update/n/n"));
rc = SQLAllocStmt(conHnd, &stmtHnd);
if (rc != SQL_SUCCESS)
{
  _tprintf(_T("Failed to allocate STMT/n"));
  goto exit2;
}

// Clear Result Data
memset(resultdata, 0, sizeof(resultdata));
chunksize   = clobdatalen / 15;   // 15 times to put

// Step 1: Prepare
rc = SQLExecDirect(stmtHnd, sqlStmt2, SQL_NTS); // select
checkSQLErr(envHnd, conHnd, stmtHnd, rc);

// Step 2: Fetch
rc = SQLFetch(stmtHnd);
checkSQLErr(envHnd, conHnd, stmtHnd, rc);
for(dtsize=0, bufp = resultdata;
    dtsize < sizeof(resultdata)/sizeof(TCHAR) && rc != SQL_NO_DATA;
    dtsize += chunksize-1, bufp += chunksize-1)
{
  int len; // len should contain the space for NULL termination
  if (dtsize+chunksize<sizeof(resultdata)/sizeof(TCHAR))
    len = chunksize;
  else
    len = sizeof(resultdata)/sizeof(TCHAR)-dtsize;

  // Step 3: GetData
  rc = SQLGetData(stmtHnd,
    1,
    SQL_C_TCHAR,
    (SQLPOINTER)bufp,
    len*sizeof(TCHAR),
    &retchklen);
}
if (!_tcscmp(resultdata, clobdata))
{
  _tprintf(_T("Succeeded!!/n/n"));
}
else
{
  _tprintf(_T("Failed!!/n/n"));
}
```

E.13 パフォーマンスとチューニング

この項では、次の項目について説明します。

- ODBCプログラミングの一般的なガイドライン
- データソース構成オプション
- DATEおよびTIMESTAMPデータ型

E.13.1 ODBCプログラミングの一般的なガイドライン

ODBCアプリケーションのパフォーマンスを向上させるには、次のプログラミング・ガイドラインを適用してください。

- アプリケーションがデータソースに頻繁に接続したり切断する場合は、接続プーリングを使用可能にします。プールされた接続の再利用は、接続の再確立に比べて非常に効率的です。
- 文を準備する回数を最小限にします。可能な場合は、バインド・パラメータを使用し、別のパラメータ値に対して文を再利用できるようにします。SQLExecuteごとに文を準備するよりも、1つの文を1回準備して複数回実行する方が効率的です。
- アプリケーションが取得しないことが判明している列、特にLONG列をSELECT文に含めないでください。LONG列がSELECT文に含まれていると、データベース・サーバー・プロトコルの特性上、アプリケーションが列をバインドするかSQLGetData操作を実行するかに関係なく、Oracle ODBC Driverでは列全体をフェッチする必要があるためです。
- データソースを更新しないトランザクションを実行する場合は、ODBCのSQLSetConnectAttr関数のSQL_ATTR_ACCESS_MODE属性をSQL_MODE_READ_ONLYに設定します。
- ODBCのエスケープ句を使用しない場合は、そのODBCのSQLSetConnectAttr関数またはSQLSetStmtAttr関数のSQL_ATTR_NOSCAN属性をtrueに設定します。
- 行数の多い表からデータを取得する場合は、ODBCのSQLFetch関数のかわりにODBCのSQLFetchScroll関数を使用します。

E.13.2 データソース構成オプション

この項では、次のODBCデータソース構成オプションを使用した場合のパフォーマンスへの影響を説明します。

- 結果セットの有効化

  このオプションによって、プロシージャ・コールから結果セット(RefCursorなど)を返すサポートが有効になります。デフォルトでは、結果セットを返すサポートが有効です。

  Oracle ODBC Driverでは、RefCursorパラメータが存在するかどうかを判断するために、データベース・サーバーを問い合せてプロシージャのパラメータ・セットとそのデータ型を判別する必要があります。この問合せによって、最初にプロシージャが準備完了になり実行されると、追加のネットワーク・ラウンド・トリップが発生します。

- LOBの有効化

  このオプションによって、LOBを挿入および更新するサポートが有効になります。デフォルトは有効です。

  Oracle ODBC Driverでは、LOBパラメータがあるかどうかを判断するために、データベース・サーバーを問い合せて、INSERT文またはUPDATE文の各パラメータのデータ型を判別する必要があります。この問合せによって、最初にINSERTまたはUPDATEが準備完了になり実行されると、追加のネットワーク・ラウンドトリップが発生します。

関連項目:

LOBの詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。

ノート:

LOB データ圧縮を行うと、SecureFiles を圧縮して、ディスク、入出力および REDO ログを節約できます。この場合、圧縮により領域の使用効率が上がるため、コストが低減されます。また、圧縮により入出力および REDO ログが縮小するため、SecureFiles のパフォーマンスが向上します。

LOB データを暗号化すると、データベースのセキュリティが強化されます。ランダムな読取りおよび書込みに対して暗号化データが使用可能になっている間、データのセキュリティは向上します。

データを圧縮および暗号化する場合、追加のメモリーが消費されます。

- TIMESTAMPをDATEとしてバインド

  SQL_TIMESTAMPパラメータを適切なOracle Databaseデータ型としてバインドします。このオプションをTRUEに設定すると、SQL_TIMESTAMPはOracleのDATEデータ型としてバインドされます。このオプションをFALSEに設定すると、SQL_TIMESTAMPはOracleのTIMESTAMPデータ型としてバインドされます。これがデフォルトです。

- カーソル・クローズの有効化

  ODBC関数SQLFreeStmtのSQL_CLOSEオプションは、関連するカーソルを文とともにクローズし、保留中のすべての結果を廃棄します。アプリケーションでは文を再度実行することによってカーソルを再オープンでき、SQLPrepareを再度実行する必要はありません。このオプションを使用する典型的な例は、しばらくの間アイドル状態になり、後で同じSQL文を再利用するアプリケーションの場合です。この場合、アプリケーションがアイドル状態になっている間、関連するサーバー・リソースを解放できます。

  Oracle ODBC Driverが位置するOCIでは、カーソルをクローズする機能をサポートしていません。したがって、デフォルトでは、SQL_CLOSEオプションはOracle ODBC Driverに影響を与えません。カーソルおよび関連するリソースはデータベース上でオープンしたままです。

  このオプションを有効にすると、データベース・サーバー上の関連するカーソルがクローズします。ただし、その結果、SQL文の解析コンテキストが失われます。ODBCアプリケーションは文を再度実行でき、SQLPrepareをコールする必要はありません。ただし、内部的には、Oracle ODBC Driverは文全体を準備して実行する必要があります。このオプションを有効にすると、文を1回準備して繰り返し実行するアプリケーションのパフォーマンスに大きな影響を与えます。

  このオプションは、サーバー上のリソースを解放する必要がある場合のみ有効にしてください。

- フェッチ・バッファ・サイズ

  odbc.iniファイルのフェッチ・バッファ・サイズ(FetchBufferSize)を、バイト単位で指定した値に設定します。この値は、Oracle ODBC DriverがOracle Databaseからクライントのキャッシュに1回にプリフェッチするデータの行数を決定するのに必要なメモリー量で、アプリケーション・プログラムが1回の問合せで要求する行数とは関係ありません。これによって、パフォーマンスが向上します。

  また、1回にフェッチするデータが20行未満であることが多いアプリケーションで、特に、速度の遅いネットワーク接続である場合や負荷の大きいサーバーからフェッチする場合は、アプリケーションの応答時間が改善されます。この設定値が高

すぎると、応答時間に悪影響が生じたり、大量のメモリーが消費される可能性があります。デフォルトは64,000バイトです。アプリケーションに対して最適な値を選択してください。

LONGおよびLOBデータ型がある場合、Oracle ODBC Driverがプリフェッチする行数はこのフェッチ・バッファ・サイズで決まりません。LONGおよびLOBデータ型が含まれると、パフォーマンスの向上は最小限になり、過剰にメモリーが使用される可能性があります。Oracle ODBC Driverはこのフェッチ・バッファ・サイズを無視し、LONGおよびLOBデータ型が存在する場合は一定の行数のみプリフェッチします。

関連項目:

「SQLDriverConnect関数の接続文字列の書式」

E.13.3 DATEおよびTIMESTAMPデータ型

WHERE句でデータベースのDATE列が使用され、その列に索引がある場合は、パフォーマンスに影響を与える可能性があります。たとえば:

```
SELECT * FROM EMP WHERE HIREDATE = ?
```

この例では、HIREDATE列の索引を使用して、問合せを迅速に実行できます。ただし、HIREDATEがDATE値で、Oracle ODBC Driverはこのパラメータ値をTIMESTAMPとして提供しているため、Oracle Databaseの問合せオプティマイザは変換関数を適用する必要があります。誤った結果(パラメータ値に0以外の小数秒がある場合に発生する可能性があります)を防ぐために、オプティマイザは、HIREDATE列に変換を適用するため、次のような文になります。

```
SELECT * FROM EMP WHERE TO_TIMESTAMP(HIREDATE) = ?
```

ただし、この場合はHIREDATE列で索引を使用できなくなります。かわりに、サーバーでは表の順次スキャンを実行します。したがって、表に多数の行がある場合は時間がかかる可能性があります。このような場合の次善策として、Oracle ODBC DriverにはTIMESTAMPをDATEとしてバインドする接続オプションが用意されています。このオプションを有効にすると、Oracle ODBC Driverは、SQL_TIMESTAMPパラメータをOracleのTIMESTAMPデータ型ではなくOracleのDATEデータ型としてバインドします。これによって、問合せオプティマイザはDATE列で索引を使用できます。

ノート:

このオプションは、DATE 列を TIMESTAMP 列としてバインドする Microsoft Access や類似のプログラムで使用することを目的にしています。実際の TIMESTAMP 列がある場合、またはデータが失われる可能性がある場合は使用しないでください。Microsoft Access では、主キーとして選択されたすべての列を使用してこの問合せを実行します。

E.14 エラー・メッセージ

エラーが発生すると、Oracle ODBC Driverは、システム固有のエラー番号、SQLSTATE(ODBCエラー・コード)およびエラー・メッセージを返します。ドライバは、ドライバが検出したエラーおよびOracle Databaseから返されたエラーの両方からこれらの情報を導出します。

固有のエラー

データソースでエラーが発生した場合、Oracle ODBC Driverは、Oracle Databaseから返されたシステム固有のエラーを

返します。Oracle ODBC DriverまたはDriver Managerがエラーを検出した場合、Oracle ODBC Driverはシステム固有のエラー番号として0(ゼロ)を返します。

SQLSTATE

データソースでエラーが発生した場合、Oracle ODBC Driverは、返されたシステム固有のエラーを適切なSQLSTATEにマップします。Oracle ODBC DriverまたはDriver Managerがエラーを検出した場合は、適切なSQLSTATEを生成します。

エラー・メッセージ

データソースでエラーが発生した場合、Oracle ODBC Driverは、Oracle Databaseから返されたメッセージに基づいてエラー・メッセージを返します。Oracle ODBC DriverまたはDriver Managerでエラーが発生した場合、Oracle ODBC Driverは、SQLSTATEに関連付けられたテキストに基づいてエラー・メッセージを返します。

エラー・メッセージの書式は次のとおりです。

```
[vendor] [ODBC-component] [data-source] error-message
```

大カッコ([ ])内の接頭辞によって、エラーの発生場所を識別します。次の表に、Oracle ODBC Driverから返される接頭辞の値を示します。データソースでエラーが発生した場合は、vendorおよびODBC-component接頭辞によって、データソースからエラーを受け取ったODBCコンポーネントのベンダーと名前を識別します。

エラー・ソース	**接頭辞**	**値**
ドライバ・マネージャ	[vendor]	[unixODBC]
	[ODBC-component]	[Driver Manager]
	[data-source]	適用なし
Oracle ODBC Driver	[vendor]	[ORACLE]
	[ODBC-component]	[Oracle ODBC Driver]
	[data-source]	適用なし
Oracle Database	[vendor]	[ORACLE]
	[ODBC-component]	[Oracle ODBC Driver]
	[data-source]	[Oracle OCI]

たとえば、次の書式に示すように、エラー・メッセージにOra接頭辞が付いていない場合、そのエラーはOracle ODBC Driverのエラーであることがわかります。

```
[Oracle][ODBC]Error message text here
```

また、次の書式に示すように、エラー・メッセージにOra接頭辞が付いている場合、このエラーはOracle ODBC Driverのエラーではありません。

```
[Oracle][ODBC][Ora]Error message text here
```

ノート:

エラー・メッセージに ORA-接頭辞が付いている場合でも、実際のエラーは複数のソースのいずれかで発生している場合があります。

エラー・メッセージのテキストがORA-接頭辞で始まっている場合、そのエラーの詳細はOracle Databaseのドキュメントを参照してください。

F データベースの制限

この付録では、データベースの制限について説明します。

F.1 データベースの制限

これらは、CREATE DATABASE文またはCREATE CONTROLFILE文でのパラメータのデフォルト値および最大値です。

ノート:

これらのパラメータ間の相互依存によって、許容値に影響を与える場合があります。

表F-1 CREATE CONTROLFILEおよびCREATE DATABASEのパラメータ

パラメータ	デフォルト	最大値
MAXLOGFILES	16	4220
MAXLOGMEMBERS	2	5
MAXLOGHISTORY	100	65534
MAXDATAFILES	30	65534
MAXINSTANCES	1	1055

次の表に、Oracle Databaseファイル・サイズ制限をバイト単位で示します。

表F-2 ファイル・サイズの制限

ファイル・タイプ	プラットフォーム	ファイル・サイズの制限
データ・ファイル	任意	4,194,303 と DB_BLOCK_SIZE パラメータの値の積
インポート/エクスポート・ファイルおよびSQL*Loaderファイル	Oracle Solaris、Linux、IBM AIX on POWER Systems (64-Bit)および HP-UX: 32 ビット・ファイル	2,147,483,647 バイト
インポート/エクスポート・ファイルおよび	Oracle Solaris、Linux、IBM AIX on POWER Systems (64-Bit)および HP-UX:	無制限 ノート: ファイル・サイズは、オペレーティング・システムでそ

ファイル・タイプ	プラットフォーム	ファイル・サイズの制限
SQL*Loader ファイル	64 ビット・ファイル	れぞれのファイル・システムに対して許可されている制限に従います。
制御ファイル	Oracle Solaris、Linux、IBM AIX on POWER Systems (64-Bit)および HP-UX	最大 201031680 の論理ブロック

索引

記号

A

B

C

D

E

F

G

H

I

J

L

M

O

P

R

S

T

U

V

X