Oracle® Application Server

mod_plsql ユーザーズ・ガイド

10*g*（10.1.3.1.0）

部品番号 : B31864-01

2006 年 12 月

ORACLE®

Oracle Application Server mod-plsql ユーザーズ・ガイド , 10*g*（10.1.3.1.0）

部品番号 : B31864-01

原本名 : Oracle Application Server mod_plsql User's Guide, 10*g* (10.1.3.1.0)

原本部品番号 : B28963-01

原本著者 : Peter Lubbers

原本協力者 : Pushkar Kapasi, Eric Lee, Thomas Van Raalte, and Nick Greenhalgh

目次

4 PL/SQL のパフォーマンスの最適化

例一覧

図一覧

表一覧

はじめに

このマニュアルでは、mod_plsql のインストール、設定およびメンテナンスの方法について説明します。

新機能

このリリースの mod_plsql の新機能は、次のとおりです。

- 停止中のデータベース接続の検出。詳細は、3.11.3 項「接続プールでの停止中のデータベース接続の検出」を参照してください。
- PL/SQL アプリケーションが処理前にリクエストを検証できることによる、より厳重なセキュリティ・モデルの定義。

以前のリリースの mod_plsql の新機能は、次のとおりです。

- 拡張されたキャッシュ・クリーン・アップ・アルゴリズム。キャッシュのクリーン・アップ時間を設定できます。
- 拡張された OWA パッケージ。マルチバイト・データベースからのレスポンス・データのバイト・パッキングを改善でき、データベースへのラウンドトリップが減少します。
- `htp.p` など､頻繁にコールされる OWA パッケージ API における PL/SQL ファンクション・コールのオーバーヘッドの減少。
- 拡張された mod_plsql でのパフォーマンス・ロギング。詳細は、付録 A「よくある質問」の「mod_plsql ではどのような種類のロギング機能を使用できますか。」を参照してください。
- DAD 構成でのスキーマ・パスワードの不明瞭化により強化されたセキュリティ。
- mod_plsql にアクセスするための仮想パスに `/pls` が必要であるという要件の削除。/pls は引き続き使用できますが、必須ではありません。mod_plsql は Oracle HTTP Server 構成に組み込まれているため、mod_plsql の URL は接頭辞 `/pls` で始まる必要はありません。

対象読者

このマニュアルは、mod_plsql を使用して PL/SQL ベースのデータベース・アプリケーションを World Wide Web 上に配置する Oracle Application Server 管理者を対象としています。

ドキュメントのアクセシビリティについて

オラクル社は、障害のあるお客様にもオラクル社の製品、サービスおよびサポート・ドキュメントを簡単にご利用いただけることを目標としています。オラクル社のドキュメントには、ユーザーが障害支援技術を使用して情報を利用できる機能が組み込まれています。HTML 形式のドキュメントで用意されており、障害のあるお客様が簡単にアクセスできるようにマークアップされています。標準規格は改善されつつあります。オラクル社はドキュメントをすべてのお客様がご利用できるように、市場をリードする他の技術ベンダーと積極的に連携して技術的な問題に対応しています。オラクル社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/accessibility/` を参照してください。

ドキュメント内のサンプル・コードのアクセシビリティについて

スクリーン・リーダーは、ドキュメント内のサンプル・コードを正確に読めない場合があります。コード表記規則では閉じ括弧だけを行に記述する必要があります。しかし JAWS は括弧だけの行を読まない場合があります。

外部 Web サイトのドキュメントのアクセシビリティについて

このドキュメントにはオラクル社およびその関連会社が所有または管理しない Web サイトへのリンクが含まれている場合があります。オラクル社およびその関連会社は、それらの Web サイトのアクセシビリティに関しての評価や言及は行っておりません。

Oracle サポート・サービスへの TTY アクセス

アメリカ国内では、Oracle サポート・サービスへ 24 時間年中無休でテキスト電話（TTY）アクセスが提供されています。TTY サポートについては、(800)446-2398 にお電話ください。

関連ドキュメント

詳細は、次のドキュメントを参照してください。

- Oracle Database ドキュメント・ライブラリの『Oracle Database アプリケーション開発者ガイド - 基礎編』
- 『Oracle HTTP Server 管理者ガイド』
- 『Oracle Application Server 管理者ガイド』

表記規則

このマニュアルでは次の表記規則を使用します。

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	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。

サポートおよびサービス

次の各項に、各サービスに接続するための URL を記載します。

Oracle サポート・サービス

オラクル製品サポートの購入方法、および Oracle サポート・サービスへの連絡方法の詳細は、次の URL を参照してください。

```
http://www.oracle.co.jp/support/
```

製品マニュアル

製品のマニュアルは、次の URL にあります。

```
http://otn.oracle.co.jp/document/
```

研修およびトレーニング

研修に関する情報とスケジュールは、次の URL で入手できます。

```
http://www.oracle.co.jp/education/
```

その他の情報

オラクル製品やサービスに関するその他の情報については、次の URL から参照してください。

```
http://www.oracle.co.jp
http://otn.oracle.co.jp
```

注意： ドキュメント内に記載されている URL や参照ドキュメントには、Oracle Corporation が提供する英語の情報も含まれています。日本語版の情報については、前述の URL を参照してください。

1

mod_plsql の構成

この章では、mod_plsql の設定および使用方法について説明します。この章の内容は、次のとおりです。

1.1 前提条件の確認

mod_plsql を実行する前に、次の前提条件を満たしている必要があります。

- mod_plsql で必要な PL/SQL Web Toolkit パッケージをロードするデータベースの SYS ユーザー・パスワードが必要です。
- mod_plsql を接続するデータベースが稼働している必要があります。
- Oracle HTTP Server mod_plsql は、OWA パッケージのリリース 10.1.2.0.4 とともに出荷されます。データベースにはリリース 10.1.2.0.4 以上の OWA パッケージをインストールすることをお薦めします。

1.2 必須パッケージのインストール

インストール後に、Oracle HTTP Server mod_plsql を製品には含まれていないデータベースで使用する必要がある場合は、`owaload.sql` スクリプトを使用して、追加の必須パッケージを手動でインストールする必要があります。

注意： Export ユーティリティを使用して全データベースのエクスポートを実行した場合でも、`OWALOAD.SQL` スクリプトを SYS で実行して、新規のターゲット・インスタンスに mod_plsql を再インストールする必要があります。インポート / エクスポート・メカニズムでは、SYS 内のオブジェクトはインポートされないため、PL/SQL ツールキットを SYS にインストールする必要があります。

---

1. `owaload.sql` ファイルが入っているディレクトリに移動します。このディレクトリは、*`ORACLE_HOME`*`/Apache/modpsql/owa` です。
2. SQL*Plus を使用して、SYS ユーザーで Oracle Database にログインします。
3. 次の問合せを実行して、現在インストールされている OWA パッケージのリリースを確認できます。

   ```
   select owa_util.get_version from dual;
   ```

   - 問合せに成功しても、10.1.2.0.4 より前のリリースが表示される場合は、新しいリリースの OWA パッケージをインストールすることをお薦めします。
   - 問合せに失敗した場合は、OWA パッケージがインストールされていないか、かなり古いリリースの OWA パッケージが実行されています。新しい OWA パッケージをインストールするか、新リリースにアップグレードすることをお薦めします。

   注意： 古い OWA パッケージの検出方法については、付録 A「よくある質問」の「データベースに重複してインストールされている OWA パッケージを検出してクリーン・アップするには、どうすればよいですか。」を参照してください。

   ---

4. SQL プロンプトで、次のコマンドを実行します。

   ```
   @owaload.sql log_file
   ```

表 1-1 に、必須パッケージのインストール時のパラメータを示します。

表 1-1 必須パッケージのインストール時のパラメータ

エレメント	説明
owaload.sql	PL/SQL Web Toolkit パッケージを SYS スキーマにインストールします。また、データベースの全ユーザーがアクセスできるよう、パブリック・シノニムを作成してパッケージをパブリックにします。このため、データベースごとに 1 回のインストールのみ必要です。
log_file	インストール・ログ・ファイルです。ログ・ファイルを作成するための書込み許可があることを確認してください。

5. ログ・ファイルをスキャンして、エラーがないことを確認します。

注意： `owaload` スクリプトは、データベース内にある OWA パッケージの既存のリリースをチェックし、次の場合にのみ新しいリリースをインストールします。

- OWA パッケージが存在しない場合。または
- 古いバージョンの OWA パッケージが検出された場合。データベースに最新の OWA パッケージが存在する場合または新しいバージョンがインストールされている場合、`owaload` スクリプトは何も処理を実行せず、その状況をログ・ファイルにレポートします。

---

6. 手動で再コンパイルします。

注意： OWA パッケージをインストールすると、依存オブジェクトがすべて無効になることがあります。このパッケージは最初のアクセス時に自動的に再コンパイルされますが、再インストール後に手動で再コンパイルすることをお薦めします。

インストール後に、`Select owa_util.get_version from dual;` を実行して、OWA パッケージのリリースをチェックします。表示されたリリースが 10.1.2.0.4 以上であることを確認してください。

7. 次へのパブリック・アクセスが付与されていることに注意してください。
   - OWA_CUSTOM
   - OWA
   - HTF
   - HTP
   - OWA_COOKIE
   - OWA_IMAGE
   - OWA_OPT_LOCK
   - OWA_PATTERN
   - OWA_SEC
   - OWA_TEXT
   - OWA_UTIL
   - OWA_CACHE
   - WPG_DOCLOAD

- OWA_MATCH

8. 次のパブリック・シノニムが作成されていることにも注意してください。
   - OWA_CUSTOM 用の OWA_CUSTOM
   - OWA_CUSTOM 用の OWA_GLOBAL
   - OWA 用の OWA
   - HTF 用の HTF
   - HTP 用の HTP
   - OWA_COOKIE 用の OWA_COOKIE
   - OWA_IMAGE 用の OWA_IMAGE
   - OWA_OPT_LOCK 用の OWA_OPT_LOCK
   - OWA_PATTERN 用の OWA_PATTERN
   - OWA_SEC 用の OWA_SEC
   - OWA_TEXT 用の OWA_TEXT
   - OWA_UTIL 用の OWA_UTIL
   - OWA_CUSTOM 用の OWA_INIT
   - OWA_CACHE 用の OWA_CACHE
   - WPG_DOCLOAD 用の WPG_DOCLOAD
   - OWA_MATCH 用の OWA_MATCH

1.3 PL/SQL アプリケーション用の DAD の構成

Web 対応の PL/SQL アプリケーションにアクセスするには、まず、mod_plsql 用に PL/SQL データベース・アクセス記述子（DAD）を構成します。DAD は、mod_plsql がデータベース・サーバーに接続して HTTP リクエストを実行する方法を指定する値のセットです。DAD には、接続詳細の他、データベースでの各種操作および mod_plsql 全般にとって重要な構成パラメータが含まれています。mod_plsql の構成パラメータの詳細は、『Oracle HTTP Server 管理者ガイド』の「mod_plsql」を参照してください。

1.4 mod_plsql の構成へのアクセス

mod_plsql の構成方法は、『Oracle HTTP Server 管理者ガイド』の「mod_plsql」を参照してください。

1.5 mod_plsql の診断結果

mod_plsql は、HTTP を介して PL/SQL アプリケーションを起動することのできる Oracle HTTP Server モジュールです。Oracle HTTP Server モジュールであるため、mod_plsql のロギングは Oracle HTTP Server を介して行われます。

ロギングは、構成ファイル `httpd.conf` の `LogLevel` パラメータで制御されます。通常、このファイルは次の場所にあります。

ORACLE_HOME`/Apache/Apache/conf`

`LogLevel` の有効な値は次のとおりです。

- `emerg`
- `alert`
- `crit`
- `error`
- `warn`
- `notice`
- `info`
- `debug`

この値は、これより上の値を対象に含みます。たとえば、`LogLevel` を `notice` に設定した場合、`notice`、`warn`、`error`、`crit`、`alert` および `emerg` のメッセージが記録されます。`LogLevel` の値を変更した場合は、変更を有効にするために Oracle HTTP Server を再起動する必要があります。

mod_plsql ログ・ファイルの内容

mod_plsql の診断情報の場所は、`httpd.conf` ファイルの `ErrorLog` という Oracle HTTP Server パラメータによって決定されます。このパラメータは `ErrorLog` と呼ばれていますが、このパラメータによって記述されるファイルには、エラー・メッセージ以外の内容も含めることができます。`ErrorLog` パラメータの一般的な値は次のとおりです。

ORACLE_HOME`/Apache/Apache/logs/error_log`

Oracle HTTP Server エラー・ログには、2 つのタイプの mod_plsql メッセージが表示されます。

- 標準の mod_plsql メッセージ
- パフォーマンスの mod_plsql メッセージ

標準の mod_plsql メッセージ

Oracle HTTP Server エラー・ログに示される標準の mod_plsql メッセージの例は、次のとおりです。

```
[Thu Jun 30 08:34:20 2005] [warn] mod_plsql: 'PlsqlCacheCleanupSize' is deprecated.
```

標準の mod_plsql メッセージの内容は次のとおりです。

- 日付および時間 : Thu June 30 08:34:20 2005
- メッセージ・レベル : warn
- このメッセージが mod_plsql 由来であることを示す記述 : mod_plsql
- メッセージ・テキスト : 'PlsqlCacheCleanupSize' is deprecated.

2

mod_plsql を使用したアプリケーション・データベース・アクセスの保護

この章では、最適なセキュリティのためにデータベースと PL/SQL を設定する方法について説明します。この章の内容は、次のとおりです。

関連資料： mod_plsql の構成パラメータの詳細は、『Oracle HTTP Server 管理者ガイド』を参照してください。

2.1 セキュリティに関する考慮事項

mod_plsql は、データベース・アカウントが権限を有するプロシージャへのアクセスに使用できる PL/SQL ゲートウェイです。これには PUBLIC に付与されているすべてのパッケージへのアクセス権も含まれているため、PL/SQL アプリケーション側で、権限のないパッケージまたはプロシージャへのアクセスに mod_plsql が使用されないようにする必要があります。デフォルトでは、mod_plsql がアクセスできない対象は次のとおりです。

- 重要なスキーマ。`SYS.*` など。
- 重要と認識されているが、PUBLIC でアクセスできるパッケージ。`OWA_*`、`DBMS_*` および `UTL_*` パッケージなど。
- URL に特殊文字を含むプロシージャ名。

mod_plsql を使用して直接実行されないように、特定のパッケージへのアクセスを明示的に拒否するこのデフォルト・モデルは、適切に設計された PL/SQL アプリケーションの大部分に対しては十分です。このレベルのセキュリティが PL/SQL アプリケーションには不十分である場合は、次の方法の 1 つ以上を使用して、PL/SQL アプリケーションを確実に保護することをお薦めします。

- mod_plsql での PlsqlRequestValidationFunction ディレクティブの定義
- mod_plsql での PlsqlExclusionList ディレクティブへのルールの追加
- カスタム認証での AUTHORIZE プロシージャの使用
- データベース表の保護
- SQL インジェクションからのデータベースの保護

2.1.1 mod_plsql での PlsqlRequestValidationFunction ディレクティブの定義

mod_plsql には、`PlsqlRequestValidationFunction` という DAD パラメータ・ディレクティブが用意されています。このディレクティブを使用すると、リクエストされたプロシージャのそれ以上の処理を許可または禁止できます。このディレクティブは、DAD からの実行を禁止されたパッケージまたはプロシージャ・コールをブロックして、PL/SQL アプリケーションについて厳重なセキュリティを実装する場合に役立ちます。

このパラメータによって定義されるファンクションには、次のプロトタイプが必要です。

```
boolean function_name (procedure_name IN varchar2)
```

起動時には、引数 `procedure_name` に、リクエストで実行しようとしているプロシージャの名前が含まれます。

たとえば、ブラウザからコールできるすべての PL/SQL アプリケーション・プロシージャがパッケージ **mypkg** 内にある場合、このファンクションの実装は例 2-1 に示すような単純なものになります。

例 2-1 プロシージャをブロックするリクエスト検証ファンクションの実装

```
boolean my_validation_check (procedure_name varchar2)
is
begin
    if (upper(procedure_name) like upper('myschema.mypkg%')) then
        return TRUE;
    else
        return FALSE;
    end if;
end;
```

ヒント：

- アプリケーションに属し、ブラウザからコールできるリクエストのみを許可するように、このファンクションを実装することをお薦めします。
- このファンクションは、すべてのリクエストについてコールされるため、必ずこのファンクションのパフォーマンスを高くしてください。次のような推奨事項があります。
  - Web からアクセスできるすべてのプロシージャが既知のパッケージの小さなセットに属すように、また、これらのパッケージが Web ブラウザから直接アクセスできないプロシージャを定義しないように、PL/SQL パッケージに名前を付けます。正しいネーミング規則を使用した検証ファンクションの実装は例 2-1 のようになります。
  - 実装で表検索を実行し、実行が許可されるパッケージまたはプロシージャを特定する場合、共有プールにカーソルを留めると、パフォーマンスがさらに改善されることがあります。

2.1.2 mod_plsql での PlsqlExclusionList ディレクティブへのルールの追加

mod_plsql には、`PlsqlExclusionList` という DAD 設定パラメータが用意されています。このパラメータを使用すると、特定のパターンを持つプロシージャをブラウザ URL から直接実行することを禁止できます。指定されるパターンには大 / 小文字区別がなく、任意の文字の組合せが 0 回以上発生することを示すワイルドカード・パターンの * を使用できます。ダイレクト URL からアクセスできないデフォルトのパターンは、`sys.*`、`dbms_*`、`utl_*`、`owa_util.*`、`owa_*`、`htp.*` および `htf.*` です。Oracle Application Server 10*g* リリース 2（10.1.2）からは、ユーザーが `PlsqlExclusionList` を使用して追加ルールを構成していても、デフォルトの組込み除外リストは引き続き有効です。以前のバージョンでは、`PlsqlExclusionList` ディレクティブが DAD 設定でオーバーライドされると、デフォルト設定は適用されませんでした。

mod_plsql では、このディレクティブで指定したパターンの他に、タブ、改行、一重引用符（'）、バックスラッシュ（\）などの特殊文字を含む URL も却下されます。

`PlsqlExclusionList` パラメータの詳細は、『Oracle HTTP Server 管理者ガイド』の「mod_plsql」を参照してください。

注意： `PlsqlExclusionList` ディレクティブを **#NONE#** に設定すると、すべての保護が無効になるため、アクティブな Web サイトには使用しないでください。

`dads.conf` という mod_plsql の構成ファイルに、`PlsqlExclusionList` ディレクティブを設定できます。この設定ファイルは、次のディレクトリにあります。

- （UNIX の場合）*ORACLE_HOME*`/Apache/modplsql/conf/`
- （Windows の場合）*ORACLE_HOME*`¥Apache¥modplsql¥conf`

ORACLE_HOME は、Oracle HTTP Server をインストールした場所です。

PUBLIC に付与されるユーザー定義の PL/SQL プロシージャに対して最適なセキュリティを確保するには、例 2-2 に示すように、`PlsqlExclusionList` ディレクティブを使用してユーザー設定を `dads.conf` ファイルに指定します。

例 2-2 PlsqlExclusionList ディレクティブの指定

```
PlsqlExclusionList myschema.mypackage*
```

2.1.3 カスタム認証での AUTHORIZE プロシージャの使用

カスタム認証を使用するアプリケーションでは、**AUTHORIZE** プロシージャの実装で制限付きパッケージまたはプロシージャへのアクセスを拒否できます。

2.1.4 データベース表の保護

アプリケーション・オブジェクト（表およびプロシージャ）を含むスキーマを、mod_plsql の実行対象であるスキーマと区別することが役に立つ場合があります。このようにすると、mod_plsql がユーザーとして接続するとき、そのユーザーに付与された API 以外を経由して、表およびその他のデータベース・オブジェクトに直接アクセスすることを禁止できます。シノニムを使用すると、スキーマ接頭辞を付けなくてもプロシージャは実行可能になります。

2.1.5 SQL インジェクションからのデータベースの保護

SQL インジェクション攻撃は、悪質なユーザーがバックエンド・データベースに対して実行されている SQL コマンドを変更できる場合に発生します。変更内容には、追加のデータを読み取ること、失敗するはずの妥当性チェックを成功させること、レコードを書き込むこと、またはその他の方法でアプリケーションの円滑な実行に影響を与えることなどがあります。

多くのアプリケーションがデータベース上に構築されており、多くの場合、ユーザー入力がデータベース問合せの一部として使用されています。アプリケーションのコーディングが慎重に行われていない場合、入力内容を注意深く構成することで、データベースに送信される問合せを変更できることがあります。

このような SQL インジェクション攻撃を回避するには、ユーザーの入力によって作成される動的 SQL を可能なかぎり使用しないようにします。

動的 SQL にユーザー入力を使用する必要がある場合には、動的 SQL で使用する前に、アプリケーション側で入力内容を適切にフィルタ処理する必要があります。たとえば、ユーザーが入力パラメータとして表の名前を入力する場合、基礎となるアプリケーション・コードでは、このパラメータに使用できる値を有効な表の名前のみにする必要があります。

2.2 mod_plsql を使用したユーザー認証

mod_plsql では、Oracle HTTP Server から提供される認証レベルに加えて、様々な認証レベルを提供します。Oracle HTTP Server はドキュメントや仮想パスなどを保護しますが、mod_plsql はデータベースにログインしたり、PL/SQL Web アプリケーションを実行するユーザーを保護します。

表 2-1 に示すように、様々な認証モードを有効化できます。

表 2-1 mod_plsql で使用する認証モード

認証モード	アプローチ
Basic	HTTP の Basic 認証を使用して認証が実行されます。ほとんどのアプリケーションでは、Basic 認証が使用されます。
グローバル OWA	認証は、PL/SQL Web Toolkit パッケージを含むスキーマの owa_custom.authorize プロシージャを使用して実行されます。
カスタム OWA	認証は、ユーザーのスキーマ（owa_customize.authorize）にあるパッケージとプロシージャを使用して実行されます。それが見つからない場合は、PL/SQL Web Toolkit パッケージを含むスキーマのパッケージとプロシージャが使用されます。
パッケージ別	認証は、ユーザーのスキーマ（packageName.authorize）にあるパッケージとプロシージャを使用して実行されます。
シングル・サインオン	認証は、Oracle Application Server Single Sign-On を使用して実行されます。このモードを使用するのは、アプリケーションが OracleAS Single Sign-On で動作する場合のみです。

2.2.1 Basic（データベース制御認証）

モジュール mod_plsql は、データベース・レベルでの認証をサポートしています。HTTP Basic 認証が使用されますが、この方式を使用してデータベースへのログオンを試行することで資格

証明が認証されます。認証は、次のどちらかのユーザー名とパスワードを使用して、ユーザーのデータベース・アカウントと比較検証されます。

- DAD に格納されているユーザー名とパスワード。エンド・ユーザーがログインする必要はありません。この方法は、公開の情報を提供する Web ページに便利です。
- ユーザーがブラウザの HTTP Basic 認証ダイアログ・ボックスを使用して入力するユーザー名とパスワード。ユーザーは、このダイアログ・ボックスにユーザー名とパスワードを入力する必要があります。

2.2.2 Oracle HTTP Server mod_plsql の Basic 認証モード

Oracle HTTP Server では、Basic 認証モードの仕組みが異なります。ユーザー名およびパスワードは DAD に格納する必要があります。Oracle HTTP Server では、ファイル・システム上のパスワード・ファイルに資格証明が格納されている、HTTP Basic 認証を使用します。認証は、そのファイルに指定されているユーザーと比較検証されます。

Basic 認証モード

mod_plsql は、Basic 認証をサポートしています。Oracle HTTP Server では、ユーザーの資格証明をファイル・システム上のパスワード・ファイルと照合して認証します。この機能は、`mod_auth` というモジュールによって提供されます。

2.2.3 グローバル OWA、カスタム OWA およびパッケージ別（カスタム認証）

カスタム認証を使用すると、データベース・レベルではなくアプリケーション自体でユーザーを認証できます。認証は、ユーザー記述の認証ファンクションをコールすることで実行されます。

カスタム認証は、OWA_CUSTOM を使用して行われ、動的ユーザー名およびパスワード認証とは組み合せることができません。カスタム認証には、DAD 設定ファイルに格納されている静的ユーザー名およびパスワードが必要です。mod_plsql は、この DAD ユーザー名およびパスワードを使用してデータベースにログインします。mod_plsql はログインすると、アプリケーション・レベルの PL/SQL フックをコールして、認証コントロールをアプリケーションに戻します。このコールバック・ファンクションは、アプリケーション開発者によって実装されます。コールバック・ファンクションの戻り値により、認証の成功または失敗が決まります。値 `TRUE` は成功を意味し、`FALSE` は失敗を意味します。

必要なカスタム認証の種類に応じて、認証ファンクションは様々な場所に配置できます。

- グローバル OWA の場合は、すべてのユーザーおよびプロシージャに対して同じ認証ファンクションをコールできます。
- カスタム OWA の場合は、各ユーザーおよびすべてのプロシージャに対して異なる認証ファンクションをコールできます。
- パッケージ別認証を使用すると、特定のパッケージのプロシージャまたは匿名プロシージャについてのみ、すべてのユーザーに対して認証ファンクションを実行できます。

たとえば、カスタム OWA を使用している場合、認証ファンクションは、ユーザーがパスワード **welcome** のユーザー **guest** としてログインしているかどうかを確認できます。また、ユーザーの IP アドレスをチェックしてアクセス権を判別できます。

表 2-2 にパラメータ値を示します。

表 2-2 カスタム認証モードとコールバック・ファンクション

モード	アクセス制御の有効範囲	コールバック・ファンクション
グローバル OWA	すべてのパッケージ	OWA パッケージ・スキーマ内の owa_custom.authorize。
カスタム OWA	すべてのパッケージ	ユーザーのスキーマ内、または見つからない場合は OWA パッケージ・スキーマ内の owa_custom.authorize。
パッケージ別	指定したパッケージ	ユーザーのスキーマ内の packageName.authorize。または anonymous.authorize がコールされます。

2.3 ユーザーの認証解除

動的認証（DAD にユーザー名とパスワードなし）を使用する DAD の場合、mod_plsql では PL/SQL プロシージャを介してユーザーをプログラム的にログオフ（HTTP 認証情報を消去）させることができ、すべてのブラウザ・インスタンスを終了する必要はありません。この機能は、Netscape 3.0 以上および Microsoft Internet Explorer でサポートされます。他のブラウザの場合、ユーザーが認証を解除するにはブラウザの終了操作が必要になることがあります。

ログアウトをシミュレートしてユーザーをサインオフ・ページにリダイレクトする独自のログアウト・プロシージャを作成すると、認証解除をプログラム的に実行できます。

MyLogOffProc プロシージャを次のように作成または置き換えます。

```
BEGIN
   -- Open the HTTP header
   owa_util.mime_header('text/html', FALSE, NULL);

   -- Send a cookie to logout
   owa_cookie.send('WDB_GATEWAY_LOGOUT', 'YES', path=>'/');

   -- Close the HTTP header
   owa_util.http_header_close;

   -- Generate the page
   htp.p('You have been logged off from the WEBSITE');
   htp.anchor( 'http://www.abc.com', 'click here');
   htp.p('<BR>bye');
END;
```

もう 1 つの認証解除方法は、URL の DAD の後に /logmeoff を追加することです。たとえば、次のようになります。

```
http://www.abc.com:2000/pls/myDAD/logmeoff
```

3

mod_plsql の概要

mod_plsql は、Web 上での PL/SQL ベースのデータベース・アプリケーションの配置をサポートします。mod_plsql は Oracle HTTP Server の一部であり、Oracle Application Server および Oracle Database とともに出荷されています。

Oracle HTTP Server の一部として、Web ブラウザから Web サーバーに送信された URL を解析し、適切な PL/SQL サブプログラムをコールしてブラウザ・リクエストを処理した後、生成されたレスポンスをブラウザに返すことが mod_plsql の仕事です。通常、mod_plsql は表示する HTML ページを構成して、Web ブラウザの HTTP リクエストにレスポンスを返します。mod_plsql の用途は他にもありますが、その中の 2 つを次に示します。

- クライアント・マシンと Oracle Database 間でファイルを送信します。テキスト・ファイルまたはバイナリ・ファイルをアップロードおよびダウンロードできます。
- カスタム・ユーザーの認証を Web アプリケーションで実行します。

Oracle HTTP Server へのプラグインとして、mod_plsql は、HTTP リクエストへのレスポンスでストアド・プロシージャが実行されるようにします。mod_plsql は、処理される URL ごとに、接続プールのデータベース・セッションを使用するか、または新規セッションを実行中に作成してプールします。mod_plsql が適切なデータベース PL/SQL プロシージャを URL 処理セッションで起動するには、まず、仮想パスを設定し、そのパスとデータベース・アクセス記述子（DAD）を関連付ける必要があります。

DAD は、名前付きの設定値のセットで、特定のデータベースのセッションを作成するために必要な情報と、特定のデータベース・ユーザーおよびパスワードを指定します。セッションのデータベース・サービス名およびグローバリゼーション・サポート設定（言語など）が含まれます。詳細は、3.2 項「データベース・アクセス記述子（DAD）」を参照してください。

実行時に mod_plsql によって実行されるストアド・プロシージャを開発するには、PL/SQL Web Toolkit を使用します。PL/SQL Web Toolkit は一連の PL/SQL パッケージで、これを使用すると HTTP リクエストに関する情報の取得、HTTP レスポンス・ヘッダー（HTTP ヘッダーの Cookie、コンテンツ・タイプ、MIME タイプなど）の指定、Cookie の設定、および HTML ページを作成するための標準 HTML タグの生成が可能です。詳細は、『Oracle Application Server PL/SQL Web Toolkit リファレンス』を参照してください。

この章の内容は、次のとおりです。

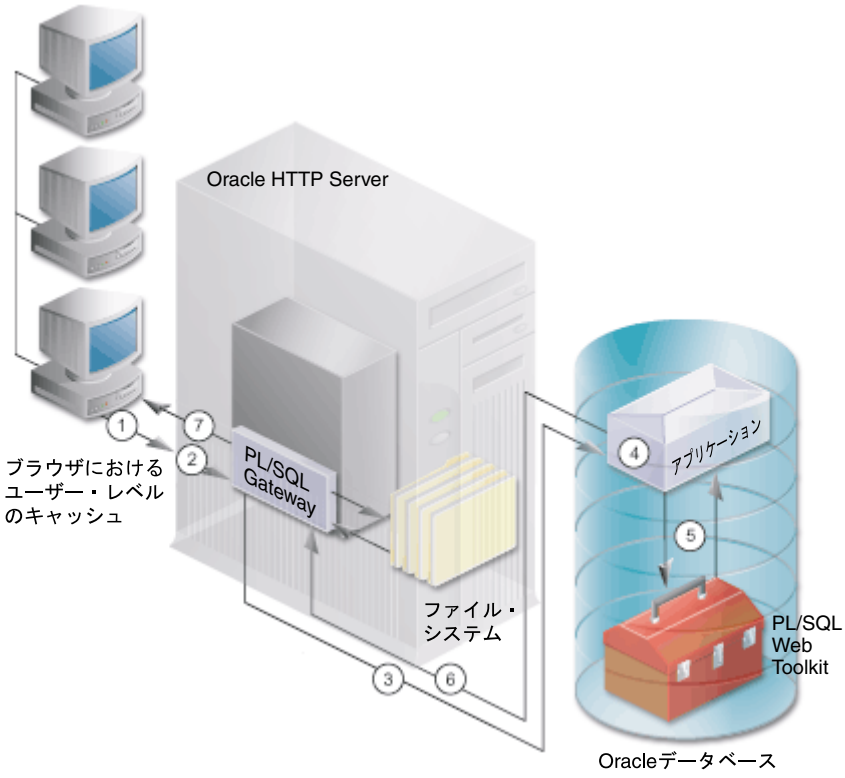
- PL/SQL ベースの Web アプリケーションでのキャッシングの使用
- mod_plsql のパフォーマンス・チューニングの領域
- mod_plsql での制限事項

3.1 クライアント・リクエストの処理

mod_plsql は、データベースと通信を行う Oracle HTTP Server のプラグインです。これによって、ブラウザ・リクエストが、SQL*Net 接続を通じてデータベース・ストアド・プロシージャ・コールにマッピングされます。通常、仮想パスの `/pls` で示されます。

次に、サーバーがクライアント・リクエストを受信するときのステップの概要（図 3-1）を説明します。

図 3-1 サーバーがクライアント・リクエストを受信する際の処理の概要



1. Oracle HTTP Server が、mod_plsql で処理されるように構成されている仮想パスを含むリクエストを受信します。
2. Oracle HTTP Server は、そのリクエストを mod_plsql にルーティングします。
3. mod_plsql は、DAD に格納された設定情報を使用して、データベースに接続します。mod_plsql は、リクエストを Oracle データベースに転送します。
4. mod_plsql は、コール・パラメータを準備して、アプリケーション内の PL/SQL プロシージャを実行します。
5. PL/SQL プロシージャは、データベースからアクセスしたデータおよび PL/SQL Web Toolkit を使用して、HTML ページを生成します。
6. レスポンスが mod_plsql に返されます。
7. Oracle HTTP Server は、そのレスポンスをクライアント・ブラウザに送信します。

mod_plsql から実行されたプロシージャは、HTTP レスポンスをクライアントに返します。この作業を簡単にするために、mod_plsql には PL/SQL Web Toolkit が含まれています。PL/SQL Web Toolkit には、OWA パッケージと呼ばれるパッケージのセットが含まれています。これらのパッケージをストアド・プロシージャ内で使用して、リクエスト情報の取得、HTML タグの作成およびクライアントへのヘッダー情報の返信を行います。すべてのユーザーがアクセスできるように、このツールキットは共通スキーマにインストールしてください。

3.2 データベース・アクセス記述子（DAD）

各 mod_plsql リクエストは、データベース・アクセス記述子（DAD）に関連付けられています。DAD は、データベース・アクセスに使用される設定値のセットです。DAD により、次のような情報が指定されます。

- データベース別名（Oracle Net サービス名）
- データベースがリモートの場合、その接続文字列
- ドキュメントのアップロードおよびダウンロード用のプロシージャ

また、DAD にはユーザー名とパスワードの情報を指定できます。指定がない場合には、URL の実行時にユーザー名とパスワードを入力するプロンプトが表示されます。

> **関連資料：** DAD パラメータの説明と mod_plsql 構成ファイルの概要は、『Oracle HTTP Server 管理者ガイド』を参照してください。

3.3 mod_plsql の実行

mod_plsql を Web ブラウザで実行するには、次の形式で URL を入力します。

protocol`://hostname[:`*port*`]/`*DAD_location*`/[[!][`*schema*`.][`*package*`.]`*proc_name*`[?`*query_string*`]]`

表 3-1 に、mod_plsql 実行時のパラメータを示します。

表 3-1 mod_plsql 実行時のパラメータ

パラメータ	説明
protocol	`http` または `https` のいずれかを使用できます。SSL の場合は、`https` を使用します。
hostname	Web サーバーが稼働しているマシンです。
port （オプション）	Web サーバーがリスニングしているポートです。指定しない場合、ポート 80 が使用されます。
DAD location	Web サーバーで設定した PL/SQL リクエストを処理するための仮想パスです。DAD location に使用できるのは ASCII 文字のみです。
! 記号 （オプション）	柔軟なパラメータの受渡しスキームを使用することを示します。詳細は、3.6.2 項「柔軟なパラメータの受渡し」を参照してください。
schema （オプション）	データベース・スキーマ名です。指定しない場合、`package.proc_name` の名前解決は、URL リクエストを処理したデータベース・ユーザーに基づいて行われます。
package （オプション）	PL/SQL ストアド・プロシージャを含んだパッケージです。指定しない場合、プロシージャはスタンドアロンになります。
proc_name	実行する PL/SQL ストアド・プロシージャです。ファンクションではなく、プロシージャを指定する必要があります。IN 引数のみ使用可能です。
?query_string （オプション）	ストアド・プロシージャのパラメータです。この文字列は、GET メソッドの書式に従います。たとえば、次のようになります。 ■ 複数のパラメータはアンパーサント（&）で区切ります。渡される値内の空白文字はプラス（+）に置き換えられます。 ■ HTML フォームを使用して文字列を生成する場合（文字列を自分で生成するのではなく）、書式化は自動的に行われます。 ■ HTTP リクエストが HTTP の POST メソッドを使用してデータを mod_plsql に送信する場合もあります。詳細は、3-6 ページの「POST メソッド、GET メソッドおよび HEAD メソッド」を参照してください。

例 3-1、例 3-2 および例 3-3 に、各種プロシージャの起動方法を示します。

例 3-1 引数をとらないプロシージャの実行

```
http://www.acme.com:9000/pls/mydad/mypackage.myproc
```

この場合、www.acme.com で稼働し、ポート 9000 でリスニングしている Web サーバーがリクエストを処理します。Web サーバーは、リクエストを受信すると、そのリクエストを mod_plsql に渡します。これは、その Web サーバーが mod_plsql を実行するように設定されていることが、/pls/mydad により示されるためです。次に、mod_plsql は、/pls/mydad に関連付けられている DAD を使用して、mypackage に格納されている myproc プロシージャを実行します。

例 3-2 引数をとるプロシージャの実行

```
http://www.acme.com:9000/pls/mydad/mypackage.myproc?a=v&b=1
```

この場合、www.acme.com で稼働し、ポート 9000 でリスニングしている Web サーバーがリクエストを処理します。Web サーバーは、リクエストを受信すると、/pls/mydad に関連付けられている DAD を使用して、mypackage に格納されている myproc プロシージャを実行し、2 つの引数 a および b（それぞれの値は v および 1）をプロシージャに渡します。

例 3-3 DAD 設定に格納されているデフォルト・プロシージャの実行

```
http://www.acme.com:9000/pls/mydad
```

この場合、www.acme.com で稼働し、ポート 9000 でリスニングしている Web サーバーがリクエストを処理します。Web サーバーは、リクエストを受信すると、/pls/mydad に関連付けられている DAD を使用して、DAD に設定されているデフォルト・プロシージャを実行します。たとえば、DAD /pls/mydad の構成パラメータ PlsqlDefaultPage が myschema.mypackage.myproc に設定されている場合、プロシージャ myschema.mypackage.myproc がリクエストに対して実行されます。

この例では、DAD 設定に指定されている mydad という DAD のデフォルトのホームページが表示されます。

POST メソッド、GET メソッドおよび HEAD メソッド

HTTP プロトコルの POST メソッド、GET メソッドおよび HEAD メソッドは、パラメータ・データをアプリケーションに渡す方法（通常は名前 / 値ペア形式）をブラウザに対して指示します。パラメータ・データは HTML フォームによって生成されます。

mod_plsql アプリケーションでは、いずれのメソッドも使用できます。各メソッドの保護レベルは、使用する転送プロトコル（HTTP または HTTPS）によって決定されます。

- POST メソッドを使用する場合、パラメータは Request Body 内で渡されます。大量のパラメータ・データをサーバーに渡す場合は、通常、POST メソッドを使用します。
- GET メソッドを使用する場合、パラメータは問合せ文字列を使用して渡されます。このメソッドには、使用するオペレーティング・システムにより、名前 / 値ペアの値の長さが環境変数の値の最大長を超えることができないという制限があります。さらに、オペレーティング・システムにより、定義できる環境変数の数も制限されます。
- HEAD メソッドを使用する場合、GET メソッドと同じ機能を使用できます。違いは、HTTP ステータス行および HTTP ヘッダーのみが返されることです。コンテンツ・データは、ブラウザに返信されません。このメソッドは、リクエストが正しく処理されているかどうかの確認のみを行うモニタリング・ツールで有効です。
- 複合モードでの使用 : mod_plsql では、一部のパラメータを問合せ文字列で渡し、それ以外のパラメータを POST データとして渡すことができます。たとえば、プロシージャ `foo`（a varchar2、b number）で、値 v と 1 をそれぞれ a と b に渡す場合は、次の 3 つの方法で値を渡して URL を作成できます。
  - すべての値を問合せ文字列の一部として指定します。

    ```
    http://host:port/pls/DAD/foo?a=v&b=1
    ```

  - すべての値を POST データの一部として指定します。

    ```
    http://host:port/pls/DAD/foo, POST data="a=v&b=1"
    ```

  - 一部のパラメータは URL で指定し、それ以外のパラメータは POST データで指定します。

    ```
    http://host:port/pls/DAD/foo?a=v, POST data="b=1"
    ```

注意 : POST データは、HTML フォーム上の入力フィールドの一部として生成されます。PL/SQL プロシージャまたは URL に、POST 文字列を手動で作成しないでください。HTML フォームの送信操作により、POST リクエストが生成され、プロシージャに値が渡されます。

3.4 トランザクション・モード

プロシージャを実行するために URL リクエストを処理した後、エラーがあれば、mod_plsql によりロールバックが実行されます。そうでない場合は、コミットが実行されます。このメカニズムでは、トランザクションが複数の HTTP リクエストにまたがることはできません。この状態を保持しないモデルの場合、アプリケーションは、通常 HTTP Cookie またはデータベース表を使用して状態を維持します。

3.5 サポートされるデータ型

HTTP でサポートされるのはキャラクタ・ストリームのみのため、mod_plsql では PL/SQL データ型の次のサブセットがサポートされます。

- NUMBER
- VARCHAR2
- TABLE OF NUMBER
- TABLE OF VARCHAR2

レコードはサポートされません。

3.6 パラメータの受渡し

mod_plsql は、次の処理をサポートしています。

- 名前によるパラメータの受渡し

  プロシージャまたはファンクションを起動する URL 内の各パラメータは、一意の名前により識別されます。パラメータのオーバーロードがサポートされています。詳細は、3.6.1 項「名前によるパラメータの受渡し（パラメータのオーバーロード）」を参照してください。

- 柔軟なパラメータの受渡し

  プロシージャの先頭に感嘆符（!）が付加されます。詳細は、3.6.2 項「柔軟なパラメータの受渡し」を参照してください。

- 大きなパラメータ（最大 32KB）の受渡し

  詳細は、3.6.3 項「大きなパラメータの受渡し」を参照してください。

注意： mod_plsql では、値を PL/SQL 表に格納することで複数の値をとる変数が処理されます。これにより、ユーザーが選択できる値の数を柔軟に設定し、ユーザーによる選択を 1 単位として簡単に処理できます。それぞれの値は、PL/SQL 表の 1 行に、索引 1 から順番に格納されます。複数の値をとる変数の最初の値（問合せ文字列に指定されている順）は索引 1、同じ変数の 2 番目の値は索引 2 に格納されます。mod_plsql によって渡される引数の順序に、PL/SQL アプリケーションが依存しないようにしてください。引数の順序は突然変わることがあるためです。PL/SQL 表の値の順序がプロシージャにとって大きな意味がある場合、PL/SQL アプリケーションを変更して、順序付けを内部的に実行する必要があります。

複数の値をとる変数がない場合、変数値はプロシージャのパラメータに位置ではなく名前で渡されるため、変数の順序は重要ではありません。

mod_plsql 環境でパラメータとして使用する PL/SQL 表は、ベース型を VARCHAR2 とする必要があります。Oracle では、VARCHAR2 を NUMBER、DATE または LONG などの他のデータ型に変換できます。VARCHAR2 変数の最大長は 32K です。

PL/SQL 表に対して 1 つ以上の値が送信されるという保証がない場合（ユーザーがオプションをまったく選択しなくてもよい場合など）は、HIDDEN フォーム・エレメントを使用して最初の値を提供します。PL/SQL 表に値を提供しないとエラーが生成されます。また、PL/SQL 表にはデフォルト値を提供できません。

3.6.1 名前によるパラメータの受渡し（パラメータのオーバーロード）

オーバーロードにより、名前が同じでパラメータの数、順序またはデータ型のファミリが異なる複数のサブプログラム（プロシージャまたはファンクション）を使用できます。オーバーロードされたサブプログラムをコールすると、PL/SQL コンパイラは渡されたデータ型に基づいて、どのサブプログラムをコールするかを決定します。

PL/SQL では、ローカル・サブプログラムとパッケージ・サブプログラムをオーバーロードできます。スタンドアロン・サブプログラムはオーバーロードできません。

パラメータ数が同じサブプログラムをオーバーロードする場合は、パラメータに違う名前を付ける必要があります。HTML データはデータ型に関連付けられないため、mod_plsql は、どのバージョンのサブプログラムをコールすればよいか判断できません。

たとえば、PL/SQL では、次の例に示すように、プロシージャに対して同じパラメータ名を使用して 2 つのプロシージャを定義できますが、これを mod_plsql で使用するとエラーが発生します。

```
-- legal PL/SQL, but not for mod_plsql
CREATE PACKAGE my_pkg AS
  PROCEDURE my_proc (val IN VARCHAR2);
  PROCEDURE my_proc (val IN NUMBER);
END my_pkg;
```

エラーを回避するには、パラメータに異なる名前を付けます。たとえば、次のようになります。

```
-- legal PL/SQL and also works for mod_plsql
CREATE PACKAGE my_pkg AS
  PROCEDURE my_proc (valvc2 IN VARCHAR2);
  PROCEDURE my_proc (valnum IN NUMBER);
END my_pkg;
```

最初のバージョンのプロシージャを実行する URL は次のようになります。

```
http://www.acme.com:9000/pls/mydad/my_pkg.my_proc?valvc2=input
```

2 番目のバージョンのプロシージャを実行する URL は次のようになります。

```
http://www.acme.com:9000/pls/mydad/my_pkg.my_proc?valnum=34
```

オーバーロードと PL/SQL 配列

パラメータ名が同じで、1 つのプロシージャのデータ型が `owa_util.ident_arr`（varchar2 の表）、もう 1 つのプロシージャのデータ型がスカラー型のオーバーロード PL/SQL プロシージャがあるとします。このような場合でも、mod_plsql はこの 2 つのプロシージャを区別できます。たとえば、次のプロシージャがあるとします。

```
CREATE PACKAGE my_pkg AS
  PROCEDURE my_proc (val IN VARCHAR2); -- scalar data type
  PROCEDURE my_proc (val IN owa_util.ident_arr); -- array data type
END my_pkg;
```

これらのプロシージャには、それぞれ `val` という同じ名前のパラメータが 1 つあります。

mod_plsql は、`val` パラメータの値を 1 つだけ持つリクエストを受け取ると、例 3-4 に示すように、スカラー・データ型のプロシージャを実行します。

例 3-4 URL 送信によるスカラー・バージョンのプロシージャの実行

次の URL を送信して、スカラー・バージョンのプロシージャを実行します。

```
http://www.acme.com:9000/pls/mydad/my_proc?val=john
```

mod_plsql は、val パラメータの値が複数存在するリクエストを受け取ると、例 3-5 に示すように、配列データ型のプロシージャを実行します。

例 3-5 URL 送信による配列バージョンのプロシージャの実行

次の URL を送信して、配列バージョンのプロシージャを実行します。

```
http://www.acme.com:9000/pls/mydad/my_proc?val=john&val=sally
```

確実に配列バージョンを実行できるようにする場合は、HTML ページで HIDDEN フォーム・エレメントを使用してダミーの値を送信します。このダミーの値は、プロシージャ内でチェックされ、破棄されます。

3.6.2 柔軟なパラメータの受渡し

ユーザーがエレメントを必要な数だけ選択できるような HTML フォームを使用できます。これらのエレメントにそれぞれ異なる名前が付いている場合は、オーバーロード・プロシージャを作成して可能な組合せを個別に処理する必要があります。または、問合せ文字列に含まれる名前が、ユーザーが選択するエレメントに関係なく確実に毎回一貫したものになるように、HIDDEN フォーム・エレメントを挿入できます。mod_plsql は、ユーザーが任意の数のエレメントを選択できる HTML フォームを処理するために、柔軟なパラメータの受渡しをサポートすることで、この操作を容易にします。

URL ベースのプロシージャの実行で柔軟なパラメータの受渡しを使用するには、URL 内でプロシージャ名の先頭に感嘆符（!）を付加します。2 パラメータまたは 4 パラメータを使用できます。2 パラメータ・インタフェースによって、mod_plsql のパフォーマンスが改善されます。4 パラメータ・インタフェースは、互換性維持のためにサポートされています。

3.6.2.1 2 パラメータ・インタフェース

```
procedure [proc_name]
     (name_array IN [array_type],
     value_array IN  [array_type]);
```

表 3-2 に、2 パラメータ・インタフェースのパラメータを示します。

表 3-2 2 パラメータ・インタフェースのパラメータ

パラメータ	説明
proc_name （必須）	実行する PL/SQL プロシージャの名前です。
name_array	問合せ文字列から取り出される名前（1 から順番に索引付けされる）を送信された順序で示します。
value_array	問合せ文字列から取り出される値（1 から順番に索引付けされる）を送信された順序で示します。
array_type （必須）	varchar2 型（owa.vc_arr など）の表による任意の PL/SQL 索引です。

例 3-6 に、2 パラメータ・インタフェースの使用について示します。

例 3-6 2 パラメータ・インタフェース

次の URL を送信します。

```
http://www.acme.com:9000/pls/mydad/!scott.my_proc?x=john&y=10&z=doe
```

感嘆符（!）接頭辞により、柔軟なパラメータの受渡しを使用することが mod_plsql に指示されます。これにより、プロシージャ scott.myproc が実行され、次の 2 つの引数が渡されます。

```
name_array ==> ('x', 'y', 'z')
value_array ==> ('john', '10', 'doe')
```

注意： このスタイルの柔軟なパラメータの受渡しを使用する場合は、パラメータ name_array および value_array を使用してプロシージャを定義する必要があります。これらの引数のデータ型は、例に示したデータ型と一致する必要があります。

3.6.2.2 4 パラメータ・インタフェース

4 パラメータ・インタフェースは、互換性維持のためにサポートされています。

```
procedure [proc_name]
     (num_entires IN NUMBER,
     name_array  IN  [array_type],
     value_array IN  [array_type],
     reserved in [array_type]);
```

表 3-3 に、4 パラメータ・インタフェースのパラメータを示します。

表 3-3 4 パラメータ・インタフェースのパラメータ

パラメータ	説明
proc_name （必須）	実行する PL/SQL プロシージャの名前です。
num_entries	問合せ文字列内の名前 / 値ペアの数です。
name_array	問合せ文字列から取り出される名前（1 から順番に索引付けされる）を送信された順序で示します。
value_array	問合せ文字列から取り出される値（1 から順番に索引付けされる）を送信された順序で示します。
reserved	未使用です。今後使用するために予約されています。
array_type （必須）	varchar2 型（owa.vc_arr など）の表による任意の PL/SQL 索引です。

例 3-7 に、4 パラメータ・インタフェースの使用について示します。

例 3-7 4 パラメータ・インタフェース

query_string で名前 x が重複している次の URL を送信します。

```
http://www.acme.com:9000/pls/mydad/!scott.my_pkg.my_proc?x=a&y=b&x=c
```

感嘆符（!）接頭辞により、柔軟なパラメータの受渡しを使用することが mod_plsql に指示されます。これにより、プロシージャ scott.my_pkg.myproc が実行され、次の 2 つの引数が渡されます。

```
num_entries ==> 3
name_array ==> ('x', 'y', 'x');
value_array ==> ('a', 'b', 'c')
reserved ==> ()
```

注意： このスタイルの柔軟なパラメータの受渡しを使用する場合は、パラメータ num_entries、name_array、value_array および reserved を使用してプロシージャを定義する必要があります。これらの引数のデータ型は、例に示したデータ型と一致する必要があります。

3.6.3 大きなパラメータの受渡し

スカラー引数として渡される値と、varchar2 引数の索引付き表の要素として渡される値に使用できるサイズは、最大 32KB です。

たとえば、柔軟なパラメータの受渡し（3.6.2 項「柔軟なパラメータの受渡し」を参照）を使用する場合、URL の query_string 部分の名前または値は、それぞれ実行されるプロシージャの name_array または value_array 引数のエレメントとして渡されます。これらの名前または値に使用できるサイズは、最大 32KB です。

3.7 ファイルのアップロードとダウンロード

mod_plsql では、次の機能を提供します。

- キャラクタ・セットを変換せずに、ファイルをロー・バイト・ストリームとしてアップロードおよびダウンロードすることが可能です。ファイルは、ドキュメント表にアップロードされます。PL/SQL アップロード・ハンドラ・ルーチンが適切な表の列を取得できるよう、主キーが渡されます。
- 異なるアプリケーションのアップロード対象ファイルを混同しないよう、DAD ごとにドキュメント表を 1 つ指定することが可能です。また、バイナリ・ラージ・オブジェクト（BLOB）のダイレクト・ダウンロード機能を使用すると、データベース表からコンテンツをダウンロードできます。
- 問合せ文字列を使用しない形式の URL 経由でこれらの表のファイルにアクセスすることが可能です。たとえば、次のようになります。

  ```
  http://www.acme.com:9000/pls/mydad/docs/cs250/lecture1.htm
  ```

  これは、URL の相互参照を含んだファイル・セットのアップロードをサポートするために必要です。
- フォーム送信ごとに複数のファイルをアップロードすることが可能です。
- ドキュメント表の LONG RAW および BLOB 型の列にファイルをアップロードすることが可能です。

この項の内容は、次のとおりです。

3.7.1 ドキュメント表の定義

DAD ごとにドキュメント格納表を指定できます。ドキュメント格納表には、次の定義が必要です。

```
CREATE TABLE [table_name] (
     NAME           VARCHAR2(256) UNIQUE NOT NULL,
     MIME_TYPE      VARCHAR2(128),
     DOC_SIZE       NUMBER,
     DAD_CHARSET    VARCHAR2(128),
     LAST_UPDATED   DATE,
     CONTENT_TYPE   VARCHAR2(128),
     [content_column_name] [content_column_type]
     [ , [content_column_name] [content_column_type]]
);
```

table_name は、ユーザーが選択できます。content_column_type 型には、LONG RAW または BLOB のいずれかを使用します。

content_column_name は、対応する content_column_type によって異なります。

- content_column_type が LONG RAW の場合、content_column_name は CONTENT にする必要があります。
- content_column_type が BLOB の場合、content_column_name は BLOB_CONTENT にする必要があります。

次に、有効なドキュメント表の定義例を示します。

```
CREATE TABLE MYDOCTABLE (
  NAME               VARCHAR(256)   UNIQUE NOT NULL,
  MIME_TYPE          VARCHAR(128),
  DOC_SIZE           NUMBER,
  DAD_CHARSET        VARCHAR(128),
  LAST_UPDATED       DATE,
  CONTENT_TYPE       VARCHAR(128),
  CONTENT            LONG RAW,
  BLOB_CONTENT       BLOB ;
);
```

3.7.1.1 CONTENT 列のセマンティクス

表の内容は、CONTENT 列に格納されます。ドキュメント表には、複数の CONTENT 列を含めることが可能です。ただし、ドキュメント表の各行について、CONTENT 列は 1 つのみ使用されます。その他の CONTENT 列は NULL に設定されます。

3.7.1.2 CONTENT_TYPE 列のセマンティクス

CONTENT_TYPE 列は、ドキュメントが格納されている CONTENT 列を追跡するために使用されます。ドキュメントがアップロードされると、mod_plsql により、この列の値が型名に設定されます。

たとえば、ドキュメントが BLOB_CONTENT 列にアップロードされた場合、ドキュメントの CONTENT_TYPE 列には文字列 BLOB が設定されます。

3.7.1.3 LAST_UPDATED 列のセマンティクス

LAST_UPDATED 列には、ドキュメントの作成日時または最終更新日時が反映されます。ドキュメントがアップロードされると、mod_plsql により、ドキュメントの LAST_UPDATED 列にデータベース・サーバーの時間が設定されます。

その後、アプリケーションがドキュメントの内容または属性を変更する場合、LAST_UPDATED の時間も更新されるようにする必要があります。

mod_plsql では、LAST_UPDATED 列を使用して、HTTP クライアント（ブラウザ）がドキュメントのキャッシュ済バージョンを使用できるかどうかをチェックし、HTTP クライアントに結果を知らせます。これにより、ネットワークの通信量が削減され、サーバーのパフォーマンスが改善されます。

3.7.1.4 DAD_CHARSET 列のセマンティクス

DAD_CHARSET 列は、ファイルのアップロード時のキャラクタ・セット設定を追跡します。この列は、今後使用するために予約されています。

3.7.2 以前のスタイルのドキュメント表の定義

WebDB リリース 2.x より前のリリースで使用されているドキュメント・モデルとの下位互換性を維持するために、mod_plsql では、CONTENT_TYPE、DAD_CHARSET および LAST_UPDATED 列が存在しない、以前のドキュメント格納表の定義もサポートしています。

```
/* older style document table definition (DEPRECATED) */
CREATE TABLE [table_name]
(
    NAME         VARCHAR2(128),
    MIME_TYPE    VARCHAR2(128),
    DOC_SIZE     NUMBER,
    CONTENT      LONG RAW
);
```

3.7.3 ドキュメントのアップロード / ダウンロードの構成パラメータ

DAD の次の設定パラメータが、ドキュメントのアップロードまたはダウンロード操作に影響します。

- 「PlsqlDocumentTablename」
- 「PlsqlDocumentPath（ドキュメント・アクセス・パス）」
- 「PlsqlDocumentProcedure（ドキュメント・アクセス・プロシージャ）」
- 「PlsqlUploadAsLongRaw」

例 3-8 に、構成パラメータの使用とその結果について示します。

例 3-8 ドキュメントのアップロード / ダウンロードのパラメータ

DAD でこれらのパラメータが次のように設定されているとします。

```
PlsqlDocumentTablename    scott.my_document_table
PlsqlUploadAsLongRaw    html
PlsqlDocumentPath    docs
PlsqlDocumentProcedure    scott.my_doc_download_procedure
```

この場合は次のようになります。

- mod_plsql は、**scott** スキーマの **my_document_table** データベース表からデータを取得するか、この表にデータを格納します。
- .html を除くすべてのファイル拡張子は、ドキュメント表に BLOB としてアップロードされます。.html 拡張子が付いたファイルはすべて Long Raw としてアップロードされます。

- DAD location の直後に **docs** キーワードを含むすべての URL では、**scott.my_doc_download_procedure** プロシージャが実行されます。

  通常、このプロシージャは **wpg_docload.download_file** をコールして、URL 指定に基づく名前を持ったファイルのダウンロードを開始します。

次に前述の構成の単純な例を示します。

```
http://www.acme.com:9000/pls/dad/docs/index.html
```

この場合は、**scott.my_document_table** データベース表の Long Raw 列から `index.html` ファイルがダウンロードされます。アプリケーション・プロシージャは開始するファイルのダウンロードを完全に制御し、ファイル・レベルのアクセス制御とバージョニングを実装する、より複雑な `PlsqlDocumentProcedure` を柔軟に定義することに注意してください。

注意： アプリケーション定義プロシージャ **scott.my_doc_download_procedure** は引数なしで定義し、CGI 環境変数に応じてリクエストを処理する必要があります。

3.7.3.1 PlsqlDocumentTablename

`PlsqlDocumentTablename` パラメータは、この DAD によってファイルがアップロードされたときにドキュメントを格納する表を指定します。

構文：

```
PlsqlDocumentTablename  [document_table_name]
```

```
PlsqlDocumentTablename  my_documents
```

または

```
PlsqlDocumentTablename  scott.my_document_table
```

3.7.3.2 PlsqlDocumentPath（ドキュメント・アクセス・パス）

`PlsqlDocumentPath` パラメータは、ドキュメントにアクセスするためのパス・エレメントを指定します。`PlsqlDocumentPath` パラメータは、URL 内で DAD 名の後に付加されます。たとえば、ドキュメント・アクセス・パスが `docs` の場合、URL は次のようになります。

```
http://www.acme.com:9000/pls/mydad/docs/myfile.htm
```

`mydad` は DAD 名で、`myfile.htm` はファイル名です。

構文：

```
PlsqlDocumentPath  [document_access_path_name]
```

3.7.3.3 PlsqlDocumentProcedure（ドキュメント・アクセス・プロシージャ）

`PlsqlDocumentProcedure` プロシージャは、アプリケーション指定のプロシージャです。このプロシージャはパラメータがなく、ドキュメント・アクセス・パスを持つ URL リクエストを処理します。ドキュメント・アクセス・プロシージャは、ファイルをダウンロードするために、`wpg_docload.download_file(filename)` をコールします。ドキュメント・アクセス・プロシージャは、URL 指定に基づいてファイル名を認識します。たとえば、アプリケーションでこれを使用して、ファイル・レベルのアクセス制御やバージョン管理を実装することが可能です。このようなアプリケーションの例は、3.7.7 項「ファイルのダウンロード」にあります。

構文：

```
PlsqlDocumentProcedure  [document_access_procedure_name]
```

例 3-9 に、PlsqlDocumentProcedure プロシージャの使用について示します。

例 3-9 PlsqlDocumentProcedure

```
PlsqlDocumentProcedure  my_access_procedure
```

または

```
PlsqlDocumentProcedure  scott.my_pkg.my_access_procedure
```

3.7.3.4 PlsqlUploadAsLongRaw

DAD パラメータ PlsqlUploadAsLongRaw は、ファイル拡張子に基づいてファイルのアップロードを設定します。PlsqlUploadAsLongRaw DAD パラメータの値は、1 行に 1 つのエントリがあるファイル拡張子のリストです。これらの拡張子を持つファイルは、mod_plsql により、ドキュメント表の LONG RAW 型の CONTENT 列にアップロードされます。他の拡張子を持つファイルは、BLOB CONTENT 列にアップロードされます。

ファイル拡張子には、テキスト・リテラル（jpeg、gif など）またはアスタリスク（*）を使用できます。アスタリスクは、PlsqlUploadAsLongRaw 設定でリストされていない拡張子を持つすべてのファイルと一致します。

構文：

```
PlsqlUploadAsLongRaw  [file_extension]
PlsqlUploadAsLongRaw *
```

例 3-10 に示すように、

[file_extension] はファイル拡張子（ピリオド（.）の有無は関係なく、たとえば、txt または .txt が使用できます）またはワイルド・カード文字（*）です。

例 3-10 に、PlsqlUploadAsLongRaw パラメータの使用について示します。

例 3-10 PlsqlUploadAsLongRaw

```
PlsqlUploadAsLongRaw  html
PlsqlUploadAsLongRaw  txt
PlsqlUploadAsLongRaw  *
```

3.7.4 ファイルのアップロード

クライアント・マシンからデータベースにファイルを送信する場合は、次に示す内容を含んだ HTML ページを作成します。

- enctype 属性に multipart/form-data が設定され、action 属性に「アクション・プロシージャ」と呼ばれる mod_plsql プロシージャ・コールが関連付けられている FORM タグ。
- type 属性と name 属性に file が設定された INPUT エレメント。INPUT type="file" エレメントを使用すると、ユーザーはファイル・システムのファイルを参照して、そこからファイルを選択できます。

ユーザーが「**Submit**」ボタンをクリックすると、次のイベントが発生します。

1. ブラウザは、ユーザーが指定したファイルとその他のフォーム・データをサーバーにアップロードします。
2. mod_plsql は、ファイルの内容をデータベースのドキュメント格納表内に格納します。表の名前は、PlsqlDocumentTablename DAD 設定から取得されます。
3. FORM の action 属性で指定したアクション・プロシージャが、ファイルのアップロードを行わずに mod_plsql プロシージャを実行する場合と同じように実行されます。

注意： HTML ドキュメントの解析は、mod_plsql では廃止されました。以前は、HTML ファイルのアップロード時に mod_plsql を使用してその内容が解析され、HTML ドキュメントが参照している他のファイルが識別されていました。その後、この情報が表に格納されていました。表の名前は、ドキュメント表の名前に part を追加したものが使用されていました。この機能は、カスタマにとって有用でないことが判明したため、リリース 9.0.4 の mod_plsql から廃止されました。

次の例に、アップロードするファイルをユーザーがファイル・システムから選択できる HTML フォームを示します。このフォームには、ファイルに関する情報を入力できるその他のフィールドも含まれています。

```
<html>
   <head>
      <title>test upload</title>
   </head>
   <body>
   <FORM enctype="multipart/form-data"
      action="pls/mydad/write_info"
      method="POST">
      <p>Author's Name:<INPUT type="text" name="who">
      <p>Description:<INPUT type="text" name="description"><br>
      <p>File to upload:<INPUT type="file" name="file"><br>
      <p><INPUT type="submit">
   </FORM>
   </body>
</html>
```

ユーザーがフォームの「**Submit**」ボタンをクリックすると、次の処理が実行されます。

1. ブラウザにより、`INPUT type="file"` エレメントにリストされたファイルがアップロードされます。
2. 次に、`write_info` プロシージャが実行されます。
3. このプロシージャは、フォームのフィールドの情報をデータベース内の表に書き込み、ページをユーザーに返します。

注意： アクション・プロシージャでは、ユーザーにレスポンスを返す必要はありませんが、次に示すように、送信が成功したか失敗したかをユーザーに知らせるようにすることをお薦めします。

```
procedure write_info (
     who         in varchar2,
     description in varchar2,
     file        in varchar2) as
begin
     insert into myTable values (who, description, file);
     htp.htmlopen;
     htp.headopen;
     htp.title('File Uploaded');
     htp.headclose;
     htp.bodyopen;
     htp.header(1, 'Upload Status');
     htp.print('Uploaded ' || file || ' successfully');
     htp.bodyclose;
     htp.htmlclose;
end;
```

名前の競合の可能性を低減するために、ブラウザから取得したファイル名の先頭に、生成されたディレクトリ名が付加されます。フォームで指定されたアクション・プロシージャにより、この名前が変更されます。このため、たとえば `/private/minutes.txt` がアップロードされた場合、mod_plsql によって表に格納される名前は、`F9080/private/minutes.txt` となります。アプリケーションは、コールしたストアド・プロシージャの中で、この名前を変更できます。たとえば、アプリケーションにより、名前を `scott/minutes.txt` に変更できます。

関連資料： RFC 1867『Form-Based File Upload in HTML』（IETF）

3.7.5 アップロード・ファイルの属性（MIME タイプ）の指定

ストアド・プロシージャは、アップロード・ファイルの名前を変更する以外にも、他のファイル属性を変更できます。たとえば、3.7.4 項「ファイルのアップロード」に示した例のフォームでは、アップロードされるドキュメントの Multipurpose Internet Mail Extension（MIME）タイプをユーザーが入力できるフィールドとして表示できます。

MIME タイプは、write_info のパラメータとして受信できます。その場合、ドキュメント表には、ファイルのアップロード時に mod_plsql がマルチパート・フォームから解析したデフォルトの MIME タイプではなく、そのドキュメントの MIME タイプが格納されます。

3.7.6 複数のファイルのアップロード

1 回の送信で複数のファイルを送信する場合は、アップロード・フォームに複数の <INPUT type="file" name="file"> エレメントが含まれている必要があります。複数のファイルの INPUT エレメントで、name に同じ名前を定義する場合、アクション・プロシージャでパラメータ名を owa.vc_arr 型として宣言する必要があります。ファイルの INPUT エレメントに一意の名前を定義することも可能で、その場合、アクション・プロシージャでそれぞれを varchar2 として宣言する必要があります。たとえば、フォームに次のエレメントが含まれているとします。

```
<INPUT type="file" name="textfiles">
<INPUT type="file" name="textfiles">
<INPUT type="file" name="binaryfile">
```

この場合、アクション・プロシージャに次のパラメータを含める必要があります。

```
procedure handle_text_and_binary_files(textfiles IN owa.vc_arr, binaryfile IN varchar2)
```

3.7.7 ファイルのダウンロード

ファイルをデータベースにアップロードした後、次に説明する 3 つの方法で、それらのファイルをデータベースからダウンロードできます。

- wpg_docload.download_file(file_name) をコールしてファイル file_name をダウンロードする PL/SQL プロシージャを定義します。
- ドキュメント・ダウンロード用の仮想パス（PlsqlDocumentPath）を DAD 設定に定義し、ユーザー定義のプロシージャ（PlsqlDocumentProcedure）とそのパスを関連付けます。mod_plsql は、DAD 名のすぐ後ろに PlsqlDocumentPath で指定された仮想パスを検出すると、ユーザー定義のプロシージャ（PlsqlDocumentProcedure）を自動的に起動します。このプロシージャは、wpg_docload.download_file(file_name) をコールしてファイル file_name のダウンロードを開始する必要があります。追加の引数を渡さずに起動されるように、このユーザー定義のプロシージャにはプロトタイプが必要です。

  たとえば、DAD の mydad で PlsqlDocumentPath が docs として、PlsqlDocumentProcedure が myschema.pkg.process_download として構成されている場合、URL の形式が http://www.acme.com:9000/pls/mydad/docs/myfile.htm のときは常に、mod_plsql はプロシージャ myschema.pkg.process_download を実行します。

  次に、process_download の実装例を示します。

```
procedure process_download is
v_filename varchar2(255);
begin
     -- getfilepath() uses the SCRIPT_NAME and PATH_INFO cgi
     -- environment variables to construct the full path name of
     -- the file URL, and then returns the part of the path name
     -- following '/docs/'
     v_filename := getfilepath;
     select name into v_filename from plsql_gateway_doc
                    where UPPER(name) = UPPER(v_filename);
     -- now we call docload.download_file to initiate
     -- the download.
```

```
    wpg_docload.download_file(v_filename);
    exception
       when others then
          v_filename := null;
end process_download;
```

- バイナリ・ラージ・オブジェクト（BLOB）のダイレクト・ダウンロード・メカニズムを使用して、BLOB をデータベース表からダウンロードします。このためには、次に示すように、標準の HTTP ヘッダー（MIME タイプやコンテンツ長など）を返す PL/SQL プロシージャをコールし、wpg_docload.download_file(blob_name) を起動して BLOB の blob_name をダウンロードします。

  1. wpg_docload.download_file(blob) をコールするストアド・プロシージャを作成します。blob は BLOB データ型です。mod_plsql には BLOB の内容に関する情報が含まれていないため、情報を入力する必要があります。
  2. Content-Type およびその他のヘッダーを設定します。

     次の例では、プロシージャは引数の名前を使用して表から BLOB を選択し、BLOB のダイレクト・ダウンロードを開始します。

     ```
     create or replace procedure download_blob(name in varchar2) is
     myblob blob;
     begin
     ```

     – name 引数を使用して、mytable から BLOB を選択します。

       ```
       select blob_data into myblob from mytable where blob_name = name;
       ```

     – コンテンツを記述するヘッダーを設定します。

       ```
       owa_util.mime_header('text/html', FALSE);
       htp.p('Content-Length: ' || dbms_lob.getlength(myblob));
       owa_util.http_header_close;
       ```

     – BLOB のダイレクト・ダウンロードを開始します。

       ```
       wpg_docload.download_file(myblob);
       end;
       ```

       mytable 表の構造は、次のとおりです。

       ```
       create table mytable
       (
       blob_name varchar2(128),
       blob_data blob
       );
       ```

     ドキュメント・ダウンロードが開始されても、BLOB のダイレクト・ダウンロードが使用されない場合は、wpg_docload.download_file に渡される引数により、ファイル名を一意に識別してドキュメント表からダウンロードできるようにする必要があります。このようなドキュメントの内容の返信時に、mod_plsql は、そのファイル名に対するドキュメント表エントリの他の列に格納された情報に基づいて、HTTP レスポンス・ヘッダーを生成します。MIME_TYPE 列、DOC_SIZE 列および LAST_UPDATED 列はそれぞれ、レスポンス・ヘッダーの Content-Type ヘッダー、Content-Length ヘッダーおよび If-Modified-Since ヘッダーを追加するために使用されます。

注意： wpg_docload.download_file API をプロシージャからコールするたびに、ファイル・ダウンロード操作が mod_plsql により開始されます。このような操作では、プロシージャによって生成される他の HTML コンテンツは、ブラウザには渡されません。

マルチバイト・キャラクタを使用するドキュメントのダウンロード

Windows プラットフォームで mod_plsql を使用しており、マルチバイト・キャラクタ・セット（日本語または韓国語）の一部として特殊文字 0x5c を含むマルチバイト・データベースに対して実行している場合、ファイル dads.conf を変更して、アプリケーションへのアクセスに使用される DAD を編集する必要があります。このためには、次のようにします。

1. *ORACLE_HOME*/Apache/modplsql/conf/dads.conf ファイルを開きます。
2. アプリケーションのアクセスに使用される DAD を探します。
3. WindowsFileConversion Off の行をこの DAD エントリに追加します。
4. ファイルを保存します。
5. Oracle HTTP Server を再起動します。

DAD 設定を更新しないと、ファイル名に 0x5c が含まれるドキュメントのダウンロード時に障害が発生します。たとえば、次のようになります。

- OracleAS Portal で、次のダウンロード・エラーが示されます。

  ```
  Error: Document not found (WWC-46000)
  ```

- ユーザー独自の PL/SQL アプリケーションに対して mod_plsql を使用すると、ファイルのダウンロードで次のエラーが発生します。

  ```
  HTTP-404 Not Found
  ```

3.8 パスのエイリアシング（ダイレクト・アクセス URL）

パスのエイリアシングにより、mod_plsql を使用するアプリケーションは、単純な URL によるアプリケーション内のオブジェクトへの直接参照を提供できます。この機能は、ドキュメント・ダウンロード機能の提供方法を汎用化するものです。パスのエイリアシングには、DAD の次の設定パラメータを使用します。

- PlsqlPathAlias
- PlsqlPathAliasProcedure

たとえば、DAD でこれらのパラメータが次のように設定されているとします。

```
PlsqlPathAlias   myalias
PlsqlPathAliasProcedure   scott.my_path_alias_procedure
```

この場合、DAD location の直後に **myalias** キーワードを含むすべての URL で、scott.my_path_alias_procedure プロシージャが実行されます。このプロシージャは、URL 指定に基づいて適切なレスポンスを開始できます。

注意： アプリケーション定義プロシージャ scott.my_path_alias_procedure は、varchar2 型の **p_path** という引数を 1 つとるように定義する必要があります。この引数は、PlsqlPathAlias に使用されているキーワードに続くすべてを受け取ります。

たとえば、前述の構成で次の URL があるとします。

```
http://www.acme.com:9000/pls/dad/myalias/MyFolder/MyItem
```

この URL により、**scott.my_path_alias_procedure** プロシージャは **MyFolder/MyItem** 引数を受け取ります。

3.9 Common Gateway Interface（CGI）環境変数

OWA_UTIL パッケージには、CGI 環境変数の値を取得するための API が付属しています。CGI 環境変数の値は、mod_plsql によって実行されるプロシージャにコンテキストを提供します。mod_plsql は CGI によって処理されませんが、mod_plsql から実行される PL/SQL アプリケーションは、これらの CGI 環境変数にアクセス可能です。

CGI 環境変数のリストは次のとおりです。

- HTTP_AUTHORIZATION
- DAD_NAME
- DOC_ACCESS_PATH
- HTTP_ACCEPT
- HTTP_ACCEPT_CHARSET
- HTTP_ACCEPT_LANGUAGE
- HTTP_COOKIE
- HTTP_HOST
- HTTP_PRAGMA
- HTTP_REFERER
- HTTP_USER_AGENT
- PATH_ALIAS
- PATH_INFO
- HTTP_ORACLE_ECID
- DOCUMENT_TABLE
- REMOTE_ADDR
- REMOTE_HOST
- REMOTE_USER
- REQUEST_CHARSET（3.9.2.1 項「REQUEST_CHARSET CGI 環境変数」を参照）
- REQUEST_IANA_CHARSET（3.9.2.2 項「REQUEST_IANA_CHARSET CGI 環境変数」を参照）
- REQUEST_METHOD
- REQUEST_PROTOCOL
- SCRIPT_NAME
- SCRIPT_PREFIX
- SERVER_NAME
- SERVER_PORT
- SERVER_PROTOCOL

PL/SQL アプリケーションは、owa_util.get_cgi_env インタフェースを使用して CGI 環境変数の値を取得できます。

構文：

```
owa_util.get_cgi_env(param_name in varchar2) return varchar2;
```

param_name は、CGI 環境変数の名前です。param_name には大 / 小文字区別があります。

3.9.1 CGI 環境変数の追加およびオーバーライド

PlsqlCGIEnvironmentList DAD パラメータは、1 行に 1 つのエントリがある、名前 / 値ペアのリストです。任意の環境変数のオーバーライドや、新規の環境変数の追加ができます。名前が付属の環境変数（3.9 項「Common Gateway Interface（CGI）環境変数」のリストを参照）の 1 つである場合、その環境変数は指定した値でオーバーライドされます。名前が付属の環境変数のリストにない場合は、パラメータで指定された名前および値と同じ新規の環境変数がリストに追加されます。

注意： mod_plsql の設定ファイルについては、『Oracle HTTP Server 管理者ガイド』を参照してください。

パラメータの値が指定されていない場合は、Oracle HTTP Server から値が取得されます。Oracle HTTP Server の場合は、次のように指定して CGI 環境変数の DOCUMENT_ROOT を渡すことができます。

```
PlsqlCGIEnvironmentList DOCUMENT_ROOT
```

この設定パラメータから渡された新規の環境変数は、owa_util.get_cgi_env インタフェースを経由して PL/SQL アプリケーションで使用できます。

例 3-11 に、環境変数のオーバーライドが指定された PlsqlCGIEnvironmentList の使用について示します。

例 3-11 環境変数のオーバーライドが指定された PlsqlCGIEnvironmentList

```
PlsqlCGIEnvironmentList SERVER_NAME=myhost.mycompany.com
PlsqlCGIEnvironmentList REMOTE_USER=testuser
```

この例では、CGI 環境変数の SERVER_NAME と REMOTE_USER が付属の環境変数リストに含まれているため、これらの環境変数が指定の値にオーバーライドされます。

例 3-12 に、新規の環境変数が指定された PlsqlCGIEnvironmentList の使用について示します。

例 3-12 新規の環境変数が指定された PlsqlCGIEnvironmentList

```
PlsqlCGIEnvironmentList MYENV_VAR=testing
PlsqlCGIEnvironmentList SERVER_NAME=
PlsqlCGIEnvironmentList REMOTE_USER=user2
```

この例では、SERVER_NAME 変数と REMOTE_USER 変数がオーバーライドされます。SERVER_NAME 変数は、値が指定されていないため、削除されます。MYENV_VAR という新規の環境変数は、付属の環境変数のリストに含まれていないため追加されます。この環境変数には、testing という値が割り当てられます。

3.9.2 PlsqlNLSLanguage

mod_plsql では、PlsqlNLSLanguage の DAD レベルの設定により、グローバリゼーション・サポート設定が制御されます。DAD レベルで PlsqlNLSLanguage が設定されていない場合、グローバリゼーション・サポート設定には Oracle の NLS_LANG パラメータの環境設定が使用されます。このパラメータの詳細は、『Oracle HTTP Server 管理者ガイド』の「mod_plsql」を参照してください。

3.9.2.1 REQUEST_CHARSET CGI 環境変数

CGI 環境変数 REQUEST_CHARSET は、PlsqlNLSLanguage の設定に基づいて設定されます。DAD レベルで PlsqlNLSLanguage が設定されていない場合、グローバリゼーション・サポート設定には Oracle の NLS_LANG パラメータの環境設定が使用されます。

PL/SQL アプリケーションは、ファンクション・コールによってこの情報にアクセスできます。たとえば、次のようになります。

```
owa_util.get_cgi_env('REQUEST_CHARSET');
```

3.9.2.2 REQUEST_IANA_CHARSET CGI 環境変数

これは、CGI 環境変数 REQUEST_CHARSET に相当する IANA（Internet Assigned Number Authority）の環境変数です。IANA は、インターネット上のキャラクタ・セットの標準をグローバルに調整する機関です。

PL/SQL アプリケーションは、ファンクション・コールによってこの情報にアクセスできます。たとえば、次のようになります。

```
owa_util.get_cgi_env('REQUEST_IANA_CHARSET');
```

3.10 PL/SQL ベースの Web アプリケーションでのキャッシングの使用

キャッシングを使用すると、PL/SQL ベースの Web アプリケーションのパフォーマンスを向上させることができます。パフォーマンスを向上させるために、中間層の PL/SQL プロシージャによって生成された Web コンテンツをキャッシュして、データベースのワークロードを減少させることができます。

この項では、次のようなキャッシングの方式について説明します。

- 検証方式の使用 : アプリケーションは、ページが最後に提示されてから変更があったかどうかをサーバーに確認します。
- 期限方式の使用 : PL/SQL ベースの Web アプリケーションは、特定の時間間隔に基づいて、ページをキャッシュするか、または再度生成する必要があるかを決定します。
- PL/SQL ベースの Web アプリケーションでのシステム・レベルおよびユーザー・レベルのキャッシング : 検証方式または期限方式を使用している場合に有効です。キャッシングのレベルは、ページが特定のユーザーに対してキャッシュされるのか、システム内のすべてのユーザーに対してキャッシュされるのかによって決まります。

これらの方式およびレベルは、PL/SQL Web Toolkit に含まれる owa_cache パッケージを使用して実装されます。

関連資料 :『Oracle Application Server PL/SQL Web Toolkit リファレンス』

3.10.1 検証方式の使用

通常、検証方式では、ページが最後に提示されてから変更があったかどうかをサーバーに確認します。ページが変更されていない場合、キャッシュされたページがユーザーに提示されます。ページが変更されている場合、新しいコピーを取得してユーザーに提示した後、それをキャッシュします。

検証方式を使用する方法には、Last-Modified 方法および Entity Tag 方法の 2 つがあります。次の 2 つの項では、これらの方式が HTTP プロトコルでどのように使用されるかを示します。PL/SQL Gateway では HTTP プロトコルを使用しませんが、多くの同じ原則が使用されます。

3.10.1.1 Last-Modified 方法

Web ページが HTTP プロトコルを使用して生成されると、そのページには **Last-Modified** レスポンス・ヘッダーが含まれます。このヘッダーは、リクエストされたコンテンツの日付（サーバーを基準とする）を示しています。ブラウザは、この日付情報をコンテンツとともに保存します。後続のリクエストが Web ページの URL に対して作成されると、ブラウザは次の処理を行います。

1. キャッシュされているバージョンがあるかどうかを確認します。
2. 日付情報を抽出します。
3. リクエスト・ヘッダーの **If-Modified-Since** を生成します。
4. リクエストをサーバーに送信します。

キャッシュ対応のサーバーは、**If-Modified-Since** ヘッダーを探し、それをコンテンツの日付と比較します。この 2 つが一致する場合は、「HTTP/1.1 304 Not Modified」などの HTTP レスポンス・ステータス・ヘッダーが生成され、コンテンツは返されません。このステータス・コードを受け取ると、キャッシュ・エントリは有効であるため、ブラウザはそれを再利用できます。

この 2 つが一致しない場合は、「HTTP/1.1 200 OK」などの HTTP レスポンス・ヘッダーが生成され、新しいコンテンツが、新しい **Last-Modified** レスポンス・ヘッダーとともに返されます。このステータス・コードを受け取ると、ブラウザはキャッシュ・エントリを新しいコンテンツおよび新しい日付情報で置き換える必要があります。

3.10.1.2 Entity Tag 方法

HTTP プロトコルによって提供されるもう 1 つの検証方法は、**ETag**（Entity Tag）レスポンス / リクエスト・ヘッダーです。このヘッダーの値は、ブラウザに対して不明瞭な文字列です。サーバーは、この文字列をアプリケーションのタイプに基づいて生成します。この方法は、日付値のみを含めることができる **If-Modified-Since** ヘッダーよりも一般的な検証方法です。

ETag 方法は、Last Modified 方法とよく似ています。サーバーは、レスポンス・ヘッダーの一部として ETag を生成します。ブラウザは、この不明瞭なヘッダー値を返されたコンテンツとともに格納します。このコンテンツに対する次のリクエストを受信すると、ブラウザは、格納された不明瞭な値を **If-Match** ヘッダーに設定してサーバーに渡します。サーバーがこの不明瞭な値を生成しているため、何をブラウザに返すかを判別できます。その他は、前述した **Last-Modified** 検証方法とまったく同じです。

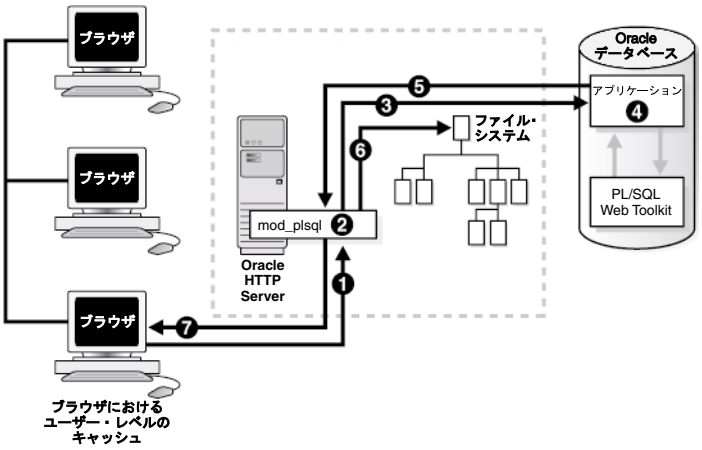
3.10.1.3 mod_plsql での検証方式の使用

HTTP 検証キャッシングをフレームワークとして使用すると、mod_plsql の検証モデルは次のようになります。

処理されるコンテンツを制御する PL/SQL ベースの Web アプリケーションでは、この種のキャッシングを使用する必要があります。この方式を使用すると、適度にパフォーマンスが改善されます。たとえば、常に変わる動的コンテンツを処理する Web アプリケーションがこれに該当します。この場合、Web アプリケーションには、処理対象に対する完全な制御が必要です。検証キャッシングでは、キャッシュされたコンテンツが古いものか、またはブラウザに返された後のものであるかを Web アプリケーションに常に確認します。

図 3-2 に、mod_plsql での検証方式の使用について示します。

図 3-2 検証方式



1. Oracle HTTP Server がクライアント・サーバーから PL/SQL プロシージャ・リクエストを受信します。Oracle HTTP Server は、そのリクエストを mod_plsql にルーティングします。
2. mod_plsql はリクエストを準備します。
3. mod_plsql は Web アプリケーションで PL/SQL プロシージャを起動し、通常の Common Gateway Interface（CGI）環境変数を Web アプリケーションに渡します。
4. PL/SQL プロシージャは、返すためのコンテンツを生成します。PL/SQL プロシージャが生成されたコンテンツをキャッシュ可能であると判断した場合、PL/SQL Web Toolkit の `owa_cache` プロシージャをコールし、タグおよびキャッシュ・レベルを設定します。

   ```
   owa_cache.set_cache(p_etag, p_level);
   ```

   表 3-4 に、検証モデルのパラメータを示します。

表 3-4 検証モデルのパラメータ

パラメータ	説明
`set_cache` プロシージャ	返されたコンテンツがキャッシュできることを mod_plsql に通知するためのヘッダーを設定します。その後、コンテンツがブラウザに返される際に、mod_plsql は、タグおよびキャッシング・レベル情報とともにコンテンツをローカル・ファイル・システムにキャッシュします。
`p_etag`	コンテンツをタグ付けするためにプロシージャが生成する文字列。
`p_level`	キャッシング・レベル：`SYSTEM`（システム・レベルの場合）または `USER`（ユーザー・レベルの場合）。

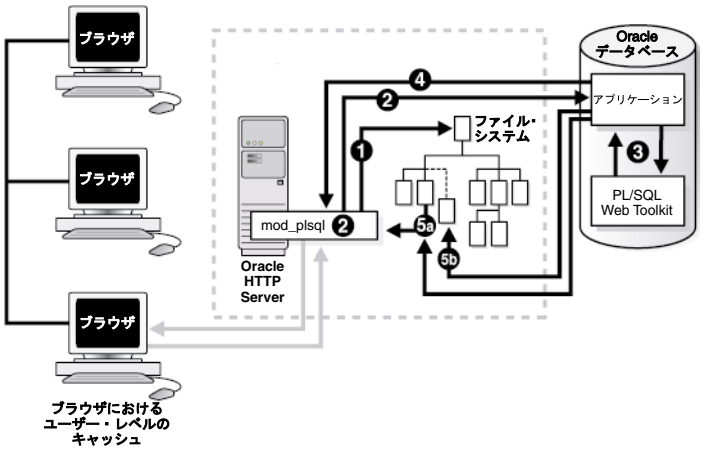
5. HTML が mod_plsql に返されます。
6. mod_plsql は、次のリクエストのために、キャッシュ可能なコンテンツをファイル・システムに格納します。
7. Oracle HTTP Server は、そのレスポンスをクライアント・ブラウザに送信します。

3.10.1.4 検証方式を使用した 2 番目のリクエスト

mod_plsql の検証方式を使用し、クライアント・ブラウザによって、同じ PL/SQL プロシージャに対する 2 番目のリクエストが作成されます。

図 3-3 に、検証方式を使用した 2 番目のリクエストを示します。

図 3-3 検証方式 - 2 番目のリクエスト



1. mod_plsql は、リクエストに対してキャッシュされたコンテンツがあることを検出します。
2. mod_plsql は、（最初のリクエストからの）同じタグおよびキャッシング・レベル情報を CGI 環境変数の一部として PL/SQL プロシージャに転送します。
3. PL/SQL プロシージャは、これらのキャッシング CGI 環境変数を使用して、コンテンツが変更されているかどうかを確認します。そのために、PL/SQL Web Toolkit の次の `owa_cache` ファンクションをコールします。

   ```
   owa_cache.get_etag;
   owa_cache.get_level;
   ```

   これらの owa ファンクションが、タグおよびキャッシング・レベルを取得します。
4. Web アプリケーションは、このキャッシング情報を mod_plsql に送信します。
5. この情報に基づいて、コンテンツを再生成する必要があるか、またはキャッシュから取得できるかを判別します。

   a. コンテンツが同じである場合は、プロシージャは `owa_cache.set_not_modified` プロシージャをコールし、コンテンツを生成しません。このため、mod_plsql ではキャッシュされたコンテンツを使用します。このキャッシュされたコンテンツは、ブラウザに直接返されます。

   b. コンテンツが変更されている場合は、新しいコンテンツを新しいタグおよびキャッシング・レベルとともに生成します。mod_plsql は古いキャッシュのコピーを新しいものに置き換え、タグおよびキャッシング・レベル情報を更新します。新しく生成されたコンテンツが、ブラウザに返されます。

3.10.2 期限方式の使用

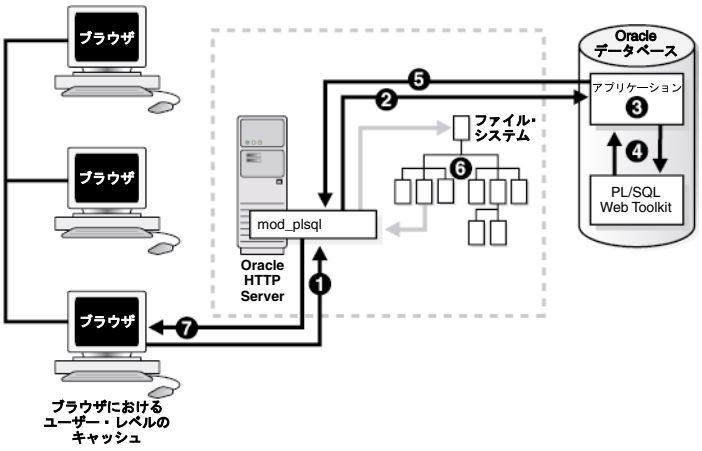
検証モデルでは、mod_plsql は、キャッシュからコンテンツを提供できるかどうかを常に PL/SQL プロシージャに確認します。期限モデルでは、プロシージャはコンテンツの有効期間をあらかじめ設定します。そのため、mod_plsql は、プロシージャに確認することなくキャッシュからコンテンツを提供できます。この方式では、データベースとのやり取りが必要ないため、パフォーマンスが一層向上します。

このキャッシング方式を使用すると、最高のパフォーマンスが得られます。PL/SQL ベースの Web アプリケーションにおいて、古いコンテンツの提供が問題にならない場合に使用します。たとえば、ニュースを毎日生成するアプリケーションがこれに該当します。ニュースは 24 時間有効に設定できます。24 時間以内であれば、アプリケーションとやり取りせずにキャッシュされたコンテンツが提供されます。これは、基本的にファイルの提供と同じです。24 時間を経過すると、mod_plsql は再度新しいコンテンツをアプリケーションから取得します。

検証モデルで説明したものと同じシナリオを考えてみます。ただし、プロシージャではキャッシングに期限モデルを使用します。

図 3-4 に、mod_plsql での期限方式の使用について示します。

図 3-4 期限方式



1. Oracle HTTP Server がクライアント・サーバーから PL/SQL Server Page リクエストを受信します。Oracle HTTP Server は、そのリクエストを mod_plsql にルーティングします。
2. mod_plsql は、リクエストを Oracle データベースに転送します。
3. mod_plsql はアプリケーションで PL/SQL プロシージャを起動し、通常の Common Gateway Interface（CGI）環境変数をアプリケーションに渡します。
4. PL/SQL プロシージャは、返すためのコンテンツを生成します。PL/SQL プロシージャが生成されたコンテンツをキャッシュ可能であると判断した場合、PL/SQL Web Toolkit の `owa_cache` プロシージャをコールし、有効期間およびキャッシュ・レベルを設定します。

   ```
   owa_cache.set_expires(p_expires, p_level);
   ```

表 3-5 に、期限モデルのパラメータを示します。

表 3-5 期限モデルのパラメータ

パラメータ	説明
set_expires プロシージャ	期限キャッシングを使用していることを mod_plsql に通知するためのヘッダーを設定します。その後、mod_plsql は、有効期間およびキャッシング・レベル情報とともにコンテンツをファイル・システムにキャッシュします。
p_expires	コンテンツが有効な分数。
p_level	キャッシング・レベル。

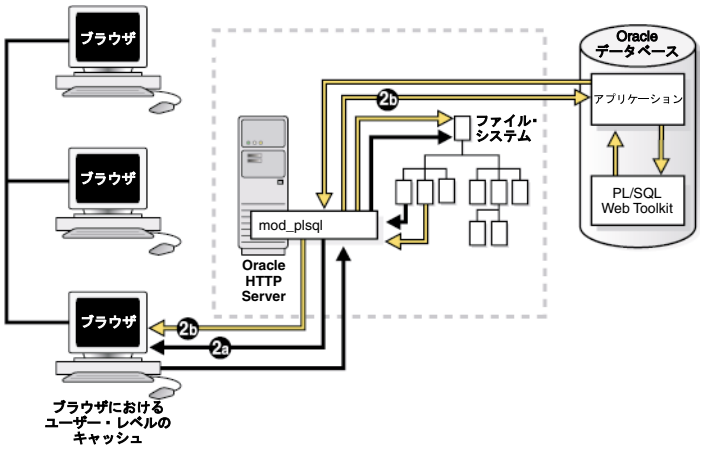
5. HTML が mod_plsql に返されます。
6. mod_plsql は、次のリクエストのために、キャッシュ可能なコンテンツをファイル・システムに格納します。
7. Oracle HTTP Server は、そのレスポンスをクライアント・ブラウザに送信します。

期限方式を使用した 2 番目のリクエスト

前述の同じ期限モデルを使用し、クライアント・ブラウザによって、同じ PL/SQL プロシージャに対する 2 番目のリクエストが作成されます。

図 3-5 に、期限方式を使用した 2 番目のリクエストを示します。

図 3-5 期限方式 - 2 番目のリクエスト



1. mod_plsql は、期限ベースの、キャッシュされたコンテンツのコピーがあることを検出します。
2. mod_plsql は、現在の時刻とこのキャッシュ・ファイルが作成された時刻の差を取得してコンテンツの有効性を確認します。
   a. この差が有効期間内にある場合は、キャッシュされたコピーはまだ新しいものであるため、データベースとやり取りせずに使用されます。このキャッシュされたコンテンツは、ブラウザに直接返されます。
   b. この差が有効期間内にない場合は、キャッシュされたコピーは古くなっています。mod_plsql は、PL/SQL プロシージャを起動して、新しいコンテンツを生成します。次に、プロシージャが期限ベースのキャッシングを再度使用するかどうかを決定します。使用する場合は、この新しいコンテンツの有効期間も決定します。新しく生成されたコンテンツが、ブラウザに返されます。

3.10.3 PL/SQL ベースの Web アプリケーションでのシステム・レベルおよびユーザー・レベルのキャッシング

PL/SQL プロシージャは、生成されたコンテンツがシステム・レベルのコンテンツか、ユーザー・レベルのコンテンツかを判断します。これにより、複数のユーザーが同じコンテンツを参照している場合、PL/SQL Gateway キャッシュによる重複ファイルの格納を減らすことができます。判断は、次の基準で行われます。

- **システム・レベル**・コンテンツの場合、プロシージャは、文字列 SYSTEM をキャッシング・レベル・パラメータとして owa_cache ファンクションに渡します（検証モデルの場合は set_cache、期限モデルの場合は set_expires）。これは、キャッシュを共有するすべてのユーザーに対して行われます。

  システム・レベル・キャッシングを使用すると、ファイル・システムの領域とシステム内のすべてのユーザーに費やす時間の両方を節約できます。たとえば、その Web アプリケーションを使用しているすべてのユーザーを対象としたコンテンツを生成する Web アプリケーションがこれに該当します。システム・レベル設定でコンテンツをキャッシングすると、コンテンツのコピーは 1 つのみファイル・システムにキャッシュされます。さらに、コンテンツはキャッシュからディレクトリに提供されるため、そのシステムのすべてのユーザーにメリットがあります。

- **ユーザー・レベル**・コンテンツの場合、プロシージャは、文字列 USER をキャッシング・レベルのパラメータとして渡します。これは、ログインしている特定のユーザーに対して行われます。格納されるキャッシュはそのユーザー独自のものです。そのユーザーのみがキャッシュを使用できます。ユーザーのタイプは、認証モードによって決まります。様々なユーザーのタイプは、表 3-6 を参照してください。

表 3-6 認証モードにより決定されるユーザーのタイプ

認証モード	ユーザーのタイプ
Single Sign-On（SSO）	軽量ユーザー
Basic	データベース・ユーザー
カスタム	リモート・ユーザー

たとえば、PL/SQL ベースの Web アプリケーションをカスタマイズしているユーザーがいない場合、出力はシステム・レベル・キャッシュに格納できます。システムのすべてのユーザーに対してキャッシュ・コピーは 1 つしかありません。複数のユーザーがキャッシュを使用できるため、ユーザー情報は使用されません。

ただし、ユーザーがアプリケーションをカスタマイズしている場合、ユーザー・レベル・キャッシュはそのユーザーに対してのみ格納されます。他のすべてのユーザーは、引き続きシステム・レベル・キャッシュを使用します。ユーザー・レベル・キャッシュ・ヒットにおいては、ユーザー情報が基準となります。ユーザー・レベル・キャッシュは、常にシステム・レベル・キャッシュをオーバーライドします。

> **関連項目：** 認証モードの詳細は、2.2 項「mod_plsql を使用したユーザー認証」を参照してください。

PL/SQL Web Toolkit のファンクション（owa_cache パッケージ）

検証方式または期限方式のどちらを使用するのかによって、コールする owa_cache のファンクションが決まります。

owa_cache パッケージには、特別なキャッシング・ヘッダーおよび環境変数を設定および取得するプロシージャが含まれます。開発者は、これらのプロシージャを使用すると、PL/SQL Gateway キャッシュをより簡単に使用できます。このパッケージは、すでにデータベースにインストールされています。

表 3-7 に、コール対象の主要なファンクションを示します。

表 3-7 主要な owa_cache のファンクション

owa ファンクション	目的
owa_cache.set_cache (p_etag IN varchar2, p_level IN varchar2)	検証モデル : ヘッダーを設定します。 ■ p_etag パラメータは、生成されたコンテンツをタグ付けします。 ■ p_level パラメータは、使用するキャッシング・レベルを表します。
owa_cache.set_not_modified	検証モデル : キャッシュされたコンテンツを使用するように mod_plsql に通知するためのヘッダーを設定します。検証ベースのキャッシュ・ヒットが発生する場合のみ使用されます。
owa_cache.get_level	検証モデル : キャッシング・レベルの USER または SYSTEM を取得します。キャッシュがヒットしない場合は NULL を返します。
owa_cache.get_etag	検証モデル : キャッシュされたコンテンツと関連付けられているタグを取得します。キャッシュがヒットしない場合は NULL を返します。
owa_cache.set_expires(p_expires IN number, p_level IN varchar2)	期限モデル : ヘッダーを設定します。 ■ p_expires パラメータは、コンテンツが有効な分数を表します。 ■ p_level パラメータは、使用するキャッシング・レベルを表します。

3.11 mod_plsql のパフォーマンス・チューニングの領域

PL/SQL ベースの Web アプリケーションのパフォーマンスを向上させるために mod_plsql をチューニングする場合、mod_plsql の内部的な動作に精通していることが重要です。この項では、mod_plsql のいくつかの機能についての概要を説明します。

この項の内容は、次のとおりです。

- mod_plsql の接続プーリング
- プールされたデータベース・セッションのクローズ
- 接続プールでの停止中のデータベース接続の検出

3.11.1 mod_plsql の接続プーリング

UNIX プラットフォームでは、mod_plsql でのデータベース・サーバー接続プーリングのロジックは、例をあげることで最もよく説明できます。次の一般的な例を考えてみます。

1. Oracle HTTP Server リスナーが起動されます。mod_plsql によって保持されている接続プールにデータベース接続はありません。
2. ブラウザが、データベース・アクセス記述子（DAD）D1 に対して mod_plsql リクエスト（R1）を作成します。
3. Oracle HTTP Server プロセスの 1 つ（`httpd` プロセス P1）が、リクエスト R1 の処理を開始します。
4. プロセス P1 の mod_plsql が接続プールを確認し、このユーザー・リクエストに対するデータベース接続がプールにないことがわかります。
5. DAD D1 の情報に基づいて、プロセス P1 の mod_plsql が、新しいデータベース接続をオープンし、PL/SQL リクエストを処理して、プールにそのデータベース接続を追加します。
6. この時点から、DAD D1 のプロセス P1 に対する後続のすべてのリクエストは、mod_plsql によってプールされたデータベース接続を使用できます。
7. DAD D1 に対するリクエストが別のプロセス（プロセス P2）で取得された場合、プロセス P2 の mod_plsql は独自のデータベース接続をオープンし、リクエストを処理して、プールにそのデータベース接続を追加します。
8. この時点から、DAD D1 のプロセス P2 に対する後続のすべてのリクエストは、mod_plsql によってプールされたデータベース接続を使用できます。
9. ここで、リクエスト R2 が DAD D2 に対して作成され、このリクエストがプロセス P1 にルーティングされるとします。
10. プロセス P1 の mod_plsql には、DAD D2 用にプールされたデータベース接続がないため、新しいデータベース・セッションが DAD D2 用に作成され、リクエストを処理した後にプールされます。この時点で、プロセス P1 には、2 つのデータベース接続がプールされています。1 つは DAD D1 用、もう 1 つは DAD D2 用です。

    ステップ 1 ～ 10 に示されたこの例で、重要な詳細は次のとおりです。

    a. 各 Oracle HTTP Server プロセスは、静的ファイル・リクエスト、サーブレット・リクエストおよび mod_plsql リクエストなど、すべてのタイプのリクエストを処理します。どの Oracle HTTP Server プロセスが次のリクエストを処理するかは制御できません。
    b. ある Oracle HTTP Server プロセスでは、別のプロセスで作成された接続プールを使用または共有できません。
    c. 各 Oracle HTTP Server プロセスは、各 DAD に対して最大で 1 つのデータベース接続をプールします。
    d. ユーザー・セッションは、DAD に対してプールされたデータベース接続内で切り替えられます。Oracle Application Server Single Sign-On（SSO）に基づく DAD の場合、プロキシ認証を使用してユーザー・セッションが切り替えられます。SSO 以外のユーザーの場合、DAD にはないユーザー名およびパスワードによる HTTP Basic 認証を使用して、ユーザーは同じ接続で再度認証されます。
    e. 複数の DAD が同じデータベース・インスタンスを指すことがありますが、データベース接続は同じプロセス内であっても DAD 間で共有されません。
    f. 未使用の DAD には、データベース接続がありません。

最悪のシナリオでは、mod_plsql によってプールされるデータベース接続の合計数は、アクティブな DAD の合計数と、1 つの Oracle HTTP Server インスタンスに対して常に実行されている Oracle HTTP Server（`httpd`）プロセスの数を掛けた数になります。Oracle HTTP Server プロセスを大きい数に設定した場合、それに相当する数のデータベース・セッションを処理するように、バックエンド・データベースを設定する必要があります。この設定値には、バック

エンド・データベースを使用する Oracle HTTP Server インスタンスの数を掛ける必要があります。

たとえば、Oracle HTTP Server インスタンスが 3 つあり、それぞれ最大 50 の `httpd` プロセスを作成するように設定されているとき、アクティブな DAD が 2 つある場合、300（`3*50*2`）セッションを処理できるようにデータベースを設定する必要があります。この数には、他の Web アプリケーションが接続するのに必要なセッションは含まれていません。

Windows プラットフォームでは、Oracle HTTP Server はシングル・プロセスとして実行されます。このようなシステムでは、mod_plsql の接続プールはスレッド間で共有され、データベース接続の合計数は、各 DAD に対する同時リクエストの数になります。データベース接続がスレッド間で共有されるため、4.3.4 項「2 リスナー方針」は Windows システムには適用されません。

3.11.2 プールされたデータベース・セッションのクローズ

プールされたデータベース・セッションは、次の状況下でクローズされます。

1. プールされた接続が、設定された数のリクエストを処理するために使用されている場合。

   デフォルトでは、mod_plsql によってプールされた各接続は、最大 1000 リクエストを処理するために使用されます。その後、データベース接続は停止され、次の mod_plsql リクエストで再確立されます。これは、PL/SQL ベースの Web アプリケーションまたは Oracle のクライアント / サーバー側でのリソース・リークがシステムに悪影響を及ぼさないようにするために行われます。デフォルト値の `1000` は、DAD 設定パラメータの `PlsqlMaxRequestsPerSession` をチューニングして変更します。

2. プールされた接続が長期間アイドル状態である場合。

   デフォルトでは、mod_plsql のクリーン・アップ・スレッドにより、プールされた各接続は、アイドル時間が 15 分を経過するとクリーン・アップされます。アイドル・セッションのタイムアウト値は、構成設定の `PlsqlIdleSessionCleanupInterval` によって制御されます。サイトで mod_plsql のコンテンツにアクセスする頻度が低いほど、アイドル・セッションのクリーン・アップはより頻繁に発生するため、ユーザーはあまり使用されないデータベース接続を確立することになります。このような状況では、パフォーマンスを向上させるために `PlsqlIdleSessionTimeoutInterval` のデフォルト設定を大きくすることを検討します。プールされたデータベース接続を長時間オープンにしておくことは、より多くのセッションで使用できるようにするために、データベースに余分な負荷が発生することを意味します。

3. UNIX システムで、Oracle HTTP Server プロセスが停止する場合。

   UNIX システムでは、Oracle HTTP Server 構成パラメータの `MaxRequestsPerChild` によって Oracle HTTP Server プロセスが停止するタイミングが制御されます。たとえば、このパラメータが `5000` に設定されている場合、各 Oracle HTTP Server プロセスは、正確に 5000 リクエストを処理した後、停止します。また、Oracle HTTP Server プロセスは、構成パラメータの `MinSpareServers`、`MaxSpareServers` および `MaxClients` に基づいて、Oracle HTTP Server のメンテナンスの一環として起動および停止することがあります。

   Oracle HTTP Server が正しく設定されていないと、Oracle HTTP Server プロセスが頻繁に起動および停止するようになり、その結果、無駄な mod_plsql の接続プーリングが発生します。最高のパフォーマンスにするには、各 Oracle HTTP Server プロセスが一定時間アクティブの状態を維持し、プロセスが停止しないように、Oracle HTTP Server を設定します。

   > **関連資料：**『Oracle HTTP Server 管理者ガイド』の「mod_plsql」を参照してください。

4. mod_plsql が接続プールで停止中の接続を検出した場合。

   詳細は、3.11.3 項「接続プールでの停止中のデータベース接続の検出」を参照してください。

3.11.3 接続プールでの停止中のデータベース接続の検出

mod_plsql は、データベースへの接続のプールを保持し、確立されたデータベース接続を後続のリクエストに再利用します。接続プールのデータベース接続からレスポンスがない場合、mod_plsql はそれを検出して、その停止中の接続を破棄し、新しいデータベース接続を後続のリクエストのために確立します。

mod_plsql の停止中のデータベース接続の検出機能により、データベース・ノードまたはインスタンスが停止したときのランダムなエラーの発生がなくなります。また、この機能は、Real Application Cluster（RAC）のような高可用性の構成では非常に有効です。ある RAC クラスタのノードが停止していると、mod_plsql はそれを検出し、他の RAC ノードを使用してリクエストの処理をすぐに開始します。

デフォルトでは、RAC ノードまたはデータベース・インスタンスが停止し、mod_plsql がノードへの接続をすでにプールしていた場合、プール内の停止中の接続を使用する最初の mod_plsql リクエストは、HTTP-503 の障害レスポンスがエンド・ユーザーに返されます。mod_plsql はこの障害を使用して、プール内のすべての停止中の接続の検出および削除をトリガーします。mod_plsql は、ノード障害が発生する前に作成されたすべての接続プールを ping します。この ping 操作は、プールされた接続を使用する次のリクエストの処理時に実行されます。ping 操作が失敗した場合、そのデータベース接続は破棄され、新しい接続が確立されて処理されます。

注意： ノード障害後、複数の mod_plsql リクエストを同時に受信し、mod_plsql がまだ最初の停止中の接続を検出していない場合、その時点では複数の障害が発生する可能性があります。

関連項目： mod_plsql リクエストがサーバーに対して作成されていない場合でも、mod_plsql が停止中のデータベース接続をクローズする他の状況の詳細は、3.11.2 項「プールされたデータベース・セッションのクローズ」を参照してください。

mod_plsql では、停止中のデータベース接続の検出機能をチューニングするための構成オプションを 2 つ提供しています。

- 停止中のデータベース接続を検出するためのオプションの指定
- タイムアウト期間の指定

3.11.3.1 停止中のデータベース接続を検出するためのオプションの指定

mod_plsql は、データベース・ノードの停止が原因と考えられる障害を検出すると接続を修正します。これは、`PlsqlConnectionValidation` パラメータによって制御されます。PlsqlConnectionValidation パラメータの詳細は、『Oracle HTTP Server 管理者ガイド』の「mod_plsql」を参照してください。

関連項目： 表 4-1「データベース・アクセス記述子（DAD）パラメータの推奨設定の概要」

3.11.3.2 タイムアウト期間の指定

PlsqlConnectionValidation パラメータが Automatic または AlwaysValidate に設定されていると、mod_plsql はプールされているデータベース接続をテストしようとします。

mod_plsql が接続プール内の不良なデータベース接続をテストするための、タイムアウト期間を指定できます。これは、PlsqlConnectionTimeout パラメータによって制御されます。このパラメータは、mod_plsql が接続が使用できないと判断するまで、テスト・リクエストが完了するのを待つ最大時間を指定します。

PlsqlConnectionTimeout パラメータの詳細は、『Oracle HTTP Server 管理者ガイド』の「mod_plsql」を参照してください。

関連項目： 表 4-1「データベース・アクセス記述子（DAD）パラメータの推奨設定の概要」

3.12 mod_plsql での制限事項

mod_plsql には、次の制限があります。

- HTTP Cookie ヘッダーの最大長は 32000 バイトです。このバイト数を超えるとエラーが発生します。
- HTTP Cookie 内にある各 Cookie の最大長は 3990 バイトです。このバイト数を超えるとエラーが発生します。この制限は、配列内の文字列の OCI 配列バインド制限によるものです。
- mod_plsql では、一度に設定できる Cookie の最大数は制限されており、変更できません。この制限は、最大数 20 に設定されています。20 を超えると、それ以降の Cookie は削除されます。
- PL/SQL Gateway は、OUT パラメータを含むプロシージャを Web インタフェースからコールすることをサポートしていません。この方法でコールすると ORA-6502 エラーが発生します。OUT 変数を含むプロシージャはコールしないことをお薦めします。ただし、現行のアーキテクチャでは、変更後の値が受渡し時の長さを超えないかぎり値を変更できます。この問題が発生する既存のアプリケーションは、次のいずれかの方法で変更する必要があります。
  - OUT パラメータを含むプロシージャがブラウザ URL 経由で直接実行されないように、この種のプロシージャのラッパーを実装します。
  - 渡されるパラメータの値が割り当てられるローカル変数を作成し、それを内部的なすべての変更に使用します。
- PL/SQL プロシージャに渡すことのできる名前 / 値ペアの合計数は 2000 です。
- mod_plsql では、1 つのプロシージャに渡すことのできる 1 つのパラメータのサイズが 32512 バイトに制限されます。
- 同じ DAD location を異なる仮想ホストで使用することはできません。
- PlsqlCacheMaxSize および PlsqlCacheTotalSize パラメータに使用できる最大値は、4294967296 バイト（4GB）です。これより大きい値を指定すると、mod_plsql で警告が発生し、内部で値が 4GB に設定されます。

4

PL/SQL のパフォーマンスの最適化

この章では、Oracle HTTP Server での PL/SQL のパフォーマンスを向上させるための方法について説明します。

この章の内容は、次のとおりです。

4.1 Oracle HTTP Server での PL/SQL のパフォーマンス - 概要

この項では、Oracle HTTP Server で PL/SQL ベースの Web アプリケーションのパフォーマンスを向上させるためのいくつかの方法について説明します。

表 4-1 に、データベース・アクセス記述子（DAD）のパラメータおよび設定の推奨事項を示します。デフォルトでは、これらの DAD パラメータは、dads.conf ファイルに指定されます。このファイルは、UNIX システムの場合、*ORACLE_HOME*/Apache/modplsql/conf ディレクトリにあります。Windows システムの場合、デフォルトでは、*ORACLE_HOME*¥Apache¥modplsql¥conf ディレクトリにあります。このディレクトリの dads.README ファイルには、DAD パラメータについて詳細に説明されています。

表 4-1 データベース・アクセス記述子（DAD）パラメータの推奨設定の概要

パラメータ	推奨設定
PlsqlAlwaysDescribeProcedure	最高のパフォーマンスにするには、off に設定します。 デフォルト値 : off
PlsqlDatabaseConnectString	新しい DAD には、ServiceNameFormat を使用します。SIDFormat は、下位互換性の場合にのみ使用します。 **注意 :** データベースの HA 構成では、接続文字列パラメータは LDAP 検索を介して解決されるように設定することをお薦めします。
PlsqlFetchBufferSize	日本語や中国語などのマルチバイト・キャラクタ・セットの場合、256 に設定すると、パフォーマンスが向上します。 デフォルト値 : 200
PlsqlIdleSessionCleanupInterval	このパラメータの値を増加すると、プールされたデータベース接続が、プール内で指定された時間、使用可能な状態を維持できます。 デフォルト値 : 15（分） **関連項目 :** 3.11.3 項「接続プールでの停止中のデータベース接続の検出」
PlsqlLogEnable	オラクル社カスタマ・サポート・センターからデバッグの目的で薦められないかぎり、Off に設定します。 デフォルト値 : off
PlsqlConnectionValidation	接続プール内で停止中の接続を検出するために mod_plsql が使用するオプションを指定します。オプションは次のとおりです。 ■ Automatic: mod_plsql は、インスタンス障害を示す障害を検出する前に作成された、すべてのプールされたデータベース接続をテストします。 ■ ThrowAwayOnFailure: mod_plsql は、インスタンス障害を示す障害を検出する前に作成された、すべてのプールされたデータベース接続を破棄します。 ■ AlwaysValidate: mod_plsql は、リクエストを発行する前に、すべてのプールされたデータベース接続を常にテストします。 **重要 :** このオプションは、各リクエストのパフォーマンス・オーバーヘッドと関連しているため、注意して使用する必要があります。 ■ NeverValidate: mod_plsql は、プールされたデータベース接続の ping を行いません。 デフォルト値 : Automatic **関連項目 :** 3.11.3.1 項「停止中のデータベース接続を検出するためのオプションの指定」

表 4-1 データベース・アクセス記述子（DAD）パラメータの推奨設定の概要（続き）

パラメータ	推奨設定
PlsqlConnectionTimeout	mod_plsql にプールされた接続のテストに対するタイムアウトを（ミリ秒単位で）指定します。 デフォルト値 : 10000 **関連項目 :** 3.11.3.2 項「タイムアウト期間の指定」
PlsqlMaxRequestsPerSession	PL/SQL ベースの Web アプリケーションがリソースまたはメモリーをリークしない場合は、このパラメータをさらに高い値（5000 など）に設定できます。 デフォルト値 : 1000 **関連項目 :** 3.11.2 項「プールされたデータベース・セッションのクローズ」および 4.3.2 項「接続プーリングのヒントおよび Oracle HTTP Server 構成」
PlsqlNLSLanguage	データベースのグローバリゼーション・サポートのパラメータと一致するように設定すると、Oracle Net Services でのキャラクタ・セット変換におけるオーバーヘッドを削減できます。
PlsqlSessionStateManagement	データベースがリリース 8.1.7.2 以上の場合、このパラメータを StatelessWithFastResetPackageState に設定します。

表 4-2 に、mod_plsql のキャッシング・オプションと、そのオプションを説明している関連項目を示します。

表 4-2 キャッシング・オプション

オプション	説明
期限方式	定期的に変わるコンテンツに対して、最高のパフォーマンスを提供します。 関連項目 : 3.10.2 項「期限方式の使用」
検証方式	不定期に変わるコンテンツに対して、良好なパフォーマンスを提供します。 関連項目 : 3.10.1 項「検証方式の使用」
システム・レベル・キャッシング	システム上のすべてのユーザーに対して 1 つのコピーをキャッシュすることにより、パフォーマンスを向上させます。 関連項目 : 3.10.3 項「PL/SQL ベースの Web アプリケーションでのシステム・レベルおよびユーザー・レベルのキャッシング」

関連資料 :

- mod_plsql のメトリックの詳細は、『Oracle Application Server パフォーマンス・ガイド』のパフォーマンス・メトリックに関する付録を参照してください。
- 表 4-1 に示された DAD パラメータの詳細は、『Oracle HTTP Server 管理者ガイド』の「mod_plsql」を参照してください。
- 『Oracle Application Server PL/SQL Web Toolkit リファレンス』

4.2 Oracle HTTP Server でのプロセス・ベースおよびスレッド・ベースの操作

この項では、Oracle HTTP Server がプロセス・ベースであるプラットフォームとスレッド・ベースであるプラットフォームである場合に当てはまる PL/SQL のパフォーマンス問題について説明します。プロセス・ベースの Oracle HTTP Server（UNIX ベースのプラットフォームで稼働するものなど）では、サーブレット、PL/SQL、静的ファイルを含め、すべてのタイプの HTTP リクエストを各プロセスで処理します。スレッド・ベースの Oracle HTTP Server（Windows ベースのプラットフォームなど）では、Oracle HTTP Server プロセスは 1 つしかなく、複数のスレッドがプロセス内にあり、個々のスレッドを使用してすべてのタイプの HTTP リクエストを処理できます。

注意： この章のいくつかの例では、PL/SQL ベースの Web アプリケーションに適用されるパフォーマンスの最適化について言及していますが、そこでは、プラットフォーム間の相違、すなわちプロセス・ベースかスレッド・ベースかが重大な意味を持ちます。

4.3 mod_plsql でのパフォーマンス・チューニング問題

mod_plsql を使用する際、パフォーマンスおよびスケーラビリティに影響を及ぼす領域は次のとおりです。

- PL/SQL ベースの Web アプリケーション開発における考慮事項とプログラミングのヒント
- 接続プーリングのヒントおよび Oracle HTTP Server 構成
- データベース・セッションの数のチューニング
- 2 リスナー方針
- オーバーヘッドの問題
- 柔軟なパラメータの受渡し（4 パラメータ）のオーバーヘッド

関連項目： 3.11 項「mod_plsql のパフォーマンス・チューニングの領域」

4.3.1 PL/SQL ベースの Web アプリケーション開発における考慮事項とプログラミングのヒント

PL/SQL Gateway ユーザーは、PL/SQL ベースの Web アプリケーションを開発する際に、次の項目について考慮する必要があります。

1. **データベース・アクセス記述子（DAD）の使用を管理します。**

   各 Oracle HTTP Server ノードで使用される DAD の数を制限してください。

   注意： 使用されていない DAD があっても、パフォーマンスに影響はありません。

   ---

2. **ネストした表を使用します。**

   PL/SQL では表を作成できます。PL/SQL 表を作成するには、表の索引およびデータ型を持つ表を作成します。表の索引は、-2147483647 ～ +2147483647 の 2 進整数です。この表索引オプションは、スパースと呼ばれ、顧客番号、従業員番号あるいはその他の便利な索引キーなど、重要な索引番号を使用できます。PL/SQL 表は、大量データの処理に使用します。

   PL/SQL には、TABLE および VARRAY（可変サイズ配列）のコレクション型があります。TABLE コレクション型は、ネストした表と呼ばれます。ネストした表は、サイズに制限がなく、スパースにできます。つまり、ネストした表内の要素は DELETE プロシージャを使用して削除できます。可変サイズ配列は、サイズに制限があり、データベースに格納され

る際に順序および添字を保持します。ネストした表のデータは、その表に関連付けられたシステム表に格納されます。可変サイズ配列は、アプリケーションがバッチ配列スタイルでデータを処理するバッチ操作に適しています。ネストした表は、ネストした表を格納表に格納して、各要素を格納表の行にマッピングすることで、問合せを効率的に行うことができます。

3. **プロシージャ名のオーバーロードを注意して使用します。**

   PL/SQL ベースの Web アプリケーションでは、プロシージャ名のオーバーロード機能の使用に注意する必要があります。複数のプロシージャに異なる名前を付けて、プロシージャ名のオーバーロードを回避することが最良の方法です。

4. **パラメータ・タイプの判別に著しいオーバーヘッドが発生するアプリケーションの再作成を検討します。**

   PL/SQL ベースの Web アプリケーションでは、パラメータ・タイプ（スカラーや配列など）の詳細が URL から十分に提供されないプロシージャの実行におけるオーバーヘッドに注意する必要があります。このような場合、プロシージャを実行する最初の試みに失敗すると、mod_plsql によって実行できるようにするためには、プロシージャ・シグネチャを記述する必要があります。

   関連項目： 4.3.5 項「オーバーヘッドの問題」

5. **2 パラメータ・スタイルの柔軟なパラメータの受渡しを利用するプロシージャを使用します。**

   プロシージャでは、4 パラメータ・スタイルのパラメータの受渡しではなく、パフォーマンスの高い 2 パラメータ・スタイルの柔軟なパラメータの受渡しを利用します。

   関連資料：

   - 『Oracle Application Server PL/SQL Web Toolkit リファレンス』
   - 4.3.6 項「柔軟なパラメータの受渡し（4 パラメータ）のオーバーヘッド」

4.3.2 接続プーリングのヒントおよび Oracle HTTP Server 構成

Oracle HTTP Server で接続プーリングを構成する際、次の項目について考慮してください。

1. デフォルトの接続プーリングおよび `PlsqlMaxRequestsPerSession` の設定値の使用

   新しいデータベース接続の作成はコストの高い操作であるため、各リクエストが独自のデータベース接続をオープンおよびクローズしないことが最も重要です。最適な方法は、あるリクエストでオープンされたデータベース接続が、後続のリクエストで確実に再利用されるようにすることです。まれなケースですが、データベースへのアクセスが非常に少なく、パフォーマンスが重要な問題ではない場合には、接続プーリングは無効化してもかまいません。たとえば、管理者がまれにサイトにアクセスして管理タスクを行う場合、管理アプリケーションへのアクセスに使用される DAD で、接続プーリングを無効化できます。接続プーリングを無効化するには、DAD パラメータ `PlsqlMaxRequestsPerSession` を値 1 に設定します。

   注意： `PlsqlMaxRequestsPerSession` を値 1 に設定すると、使用可能なデータベース・セッション数が減り、パフォーマンスに影響を及ぼすことがあります。

2. UNIX システムでは、プロセスが一度起動したら、そのプロセスが一定の時間起動した状態を維持するように、Oracle HTTP Server 構成を正しくチューニングする必要があります。そうしないと、mod_plsql の接続プーリングは有効に利用できなくなります。Oracle HTTP Server リスナーは、プロセスの起動および停止を頻繁に行わないようにする必要があります。サイトの負荷を適切に分析して、Web サイトに対する平均の負荷を特定します。Oracle HTTP Server 構成をチューニングして、`httpd` プロセスの数を、システムに対する平均の負荷を処理できるように設定する必要があります。さらに、`httpd.conf` ファ

イルの構成パラメータ MaxClients は、同様に、ランダムに発生する負荷の急激な上昇を処理できる必要があります。

3. UNIX システムでは、Oracle HTTP Server プロセスは、プロセスが最終的に停止して再起動されるように設定する必要があります。これは、Oracle HTTP Server 経由でアクセスされる様々なコンポーネントで発生する可能性があるメモリー・リークを管理するために必要です。特に、mod_plsql では、データベース・セッションのリソース・リークが問題を引き起こさないようにするために必要です。MaxRequestsPerChild 構成パラメータが高い数値に設定されていることを確認してください。PL/SQL ベースの Web アプリケーションの場合は、このパラメータを 0（ゼロ）に設定しないでください。

4. 負荷の高いサイトの場合、Oracle HTTP Server の構成パラメータ KeepAlive を無効にする必要があります。これにより、各プロセスは、現在のリクエストの処理が完了するとただちに他のクライアントからのリクエストを処理できるようになります。負荷が低く、Oracle HTTP Server プロセスの数が常に Oracle HTTP Server リスナーに対する同時リクエストの数より多いサイトの場合、KeepAlive パラメータを有効にすると、パフォーマンスは向上します。このような場合、必ず KeepAliveTimeout パラメータを適切にチューニングしてください。

5. Oracle HTTP Server 構成で Timeout の値を低くすることもできます。これにより、クライアントが適時にレスポンスを返さない場合、Oracle HTTP Server プロセスはより早く解放されます。この値を低くしすぎないでください。値が低すぎると、その値より遅いクライアントのレスポンスがタイムアウトします。

6. ほとんどの Web サイトには、一貫したユーザー・インタフェースのために各画面に表示される静的イメージ・ファイルが多数あります。このようなファイルはほとんど変わることがないため、Oracle HTTP Server リスナーにより処理される各イメージを mod_expires でタグ付けすることで、システムに対する負荷を大幅に減らすことができます。また、Oracle Application Server Web Cache で Web サイトをフロントエンドにすることも検討してください。

   - Web サイトでの mod_expires の使用が有効かどうかを確認する方法
     - Netscape またはページ・キャッシング情報を表示できる任意のブラウザを使用し、サイトで頻繁にアクセスされる Web ページをいくつか開きます。各ページでマウスを右クリックし、ポップアップ・メニューから「ページ情報を表示」（または使用しているブラウザのこれに相当するコマンド）を選択します。ページ情報ウィンドウのトップ・パネルに、様々なイメージおよび静的コンテンツが多数表示される場合、そのサイトでの mod_expires の使用は有効です。
     - また、Oracle HTTP Server アクセス・ログをチェックして、HTTP 304（Not Modified）ステータスになるリクエストの割合を確認することもできます。grep ユーティリティを使用して access_log の 304 を検索し、この結果の行数を access_log の合計の行数で割ります。この割合が高い場合、そのサイトでの mod_expires の使用は有効です。
   - 静的ファイルを Expires ヘッダーでタグ付けする方法
     - 静的イメージ・ファイルの処理に使用される Location ディレクティブを探します。ExpiresActive および ExpiresDefault ディレクティブを追加します。

```
Alias /images/ "/u01/app/oracle/myimages/"
<Directory "/u01/app/oracle/myimages/">
     AllowOverride None
     Order allow, deny
     Allow from all
     ExpiresActive On
     ExpiresDefault A2592000
</Directory>
```

     - ブラウザは、現在から 30 日間、処理されるすべての静的ファイルを /images パスからキャッシュします。詳細は、『Oracle HTTP Server 管理者ガイド』を参照してください。

- 静的ファイルが Expires ヘッダーでタグ付けされているかどうかを確認する方法
  - Netscape または任意のブラウザを使用して、ブラウザ内のすべてのキャッシュ済ファイルをクリーン・アップします。
  - Expires ヘッダーでタグ付けされたイメージがある Web ページを開きます。ページでマウスを右クリックし、ポップアップ・メニューから**「ページ情報を表示」**を選択します。または、使用しているブラウザのこれに相当するコマンドを使用します。
  - ページ情報のトップ・パネルで、Expires ヘッダーでタグ付けされているイメージを選択します。
  - 下部のパネルに表示される情報を確認します。Expires ヘッダーは有効な日付に設定されています。このエントリが「指定されていません」となっている場合、そのファイルは Expires ヘッダーでタグ付けされていません。

4.3.3 データベース・セッションの数のチューニング

データベース・セッションの数をチューニングする際、次の項目について考慮してください。

1. Oracle init$SID.ora 構成ファイルの processes および sessions パラメータは、Oracle で最大数のデータベース・セッションを処理できるように設定する必要があります。この数は、DAD の数に Oracle HTTP Server プロセスの最大数を掛け、さらに Oracle HTTP Server インスタンスの数を掛けた値に比例します。
2. 2 リスナー方針または共有サーバーを使用すると、データベース・セッションの数を減らすことができます。4.3.4 項「2 リスナー方針」を参照してください。
3. UNIX プラットフォームでは、接続プールは Oracle HTTP Server プロセス間で共有されません。このため、アプリケーションではできるだけ少ない DAD を使用することをお薦めします。

4.3.4 2 リスナー方針

Oracle HTTP Server がプロセス・ベースであるプラットフォーム（すべての UNIX ベースのプラットフォームなど）では、サーブレット、PL/SQL、静的ファイルおよび CGI を含め、すべてのタイプの HTTP リクエストを各プロセスで処理します。1 つの Oracle HTTP Server リスナー設定で、各 httpd プロセスはデータベースへの独自の接続プールを保持します。データベース・セッションの最大数は、httpd.conf 構成ファイルの StartServers、MinSpareServers および MaxSpareServers の設定およびシステムに対する負荷により決まります。このアーキテクチャでは、mod_plsql リクエストの数に基づいてデータベース・セッションの数をチューニングできません。mod_plsql リクエストの数に基づいてデータベース・セッションの数をチューニングするには、mod_plsql リクエスト専用に別の HTTP リスナーをインストールします。この方法により、mod_plsql リクエストの処理に必要なデータベース・セッションの数を大幅に減らすことができます。

たとえば、メインの Oracle HTTP Server リスナーが mylsnr1.mycompany.com のポート 7777 で稼働しているとします。まず、別の Oracle HTTP Server リスナーを mylsnr2.mycompany.com のポート 8888 にインストールします。次に、mylsnr1.mycompany.com:7777 に対して行われたすべての mod_plsql リクエストを、mylsnr2.mycompany.com:8888 の 2 番目のリスナーにリダイレクトします。手順は次のとおりです。

1. mylsnr1.mycompany.com:7777 に対するすべての PL/SQL リクエストを mylsnr2.mycompany.com:8888 にリダイレクトするには、次のように構成を変更します。
   a. ポート 7777 で稼働している Oracle HTTP Server リスナーについては、*ORACLE_HOME*/Apache/modplsql/conf/plsql.conf ファイルを編集します。行の先頭に # を挿入して、次の行をコメント化します。

      ```
      #LoadModule plsql_module...
      ```

b. mylsnr1.mycompany.com の PL/SQL リクエストの処理に使用される DAD の場所を、mylsnr2.mycompany.com の *ORACLE_HOME*/Apache/modplsql/conf/dads.conf 構成ファイルにコピーします。

行の先頭に # 文字を挿入して、mylsnr1.mycompany.com 上の DAD の場所の構成パラメータをコメント化します。

```
#<Location /pls/portal>
#...
#</Location>
```

c. 次の行を dads.conf に追加して、この DAD の場所に対するすべての mod_plsql リクエストを 2 番目のリスナーに転送するように、このリスナーを設定します。

```
ProxyPass /pls/portal http://mylsnr2.mycompany.com:8888/pls/portal
```

すべての DAD の場所について、この設定の手順を繰り返します。

2. PL/SQL プロシージャはブラウザに表示される URL を生成するため、すべての URL が mylsnr2.mycompany.com:8888 の内部 mod_plsql リスナーを参照せずに構成されていることが重要です。PL/SQL ベースの Web アプリケーションで URL がどのように生成されるかにより、次の 3 つのオプションがあります。

- URL がアプリケーションにハードコードされている場合、常にハードコードされた値（HOST=mylsnr1.mycompany.com および PORT=7777）を使用して URL が生成されていることを確認してください。この場合、変更は不要です。
- PL/SQL ベースの Web アプリケーションが常に CGI 環境変数 SERVER_NAME および SERVER_PORT を使用する場合、mylsnr2.mycompany.com のリスナーの構成を変更することは簡単です。ファイルを編集し、2 番目のリスナーの *ORACLE_HOME*/Apache/conf/httpd.conf ファイルの ServerName および Port 行を次のように変更します。

  ```
  ServerName mylsnr1.mycompany.com (was mylsnr2.mycompany.com)
  Port 7777                        (was 8888)
  ```

- URL が CGI 環境変数 HTTP_HOST を使用して生成されている場合、ポート 8888 で稼働している Oracle HTTP Server リスナーの CGI 環境変数をオーバーライドする必要があります。各 DAD の *ORACLE_HOME*/Apache/modplsql/conf/dads.conf ファイルに次の行を追加して、デフォルトの CGI 環境変数 HOST、SERVER_NAME および SERVER_PORT をオーバーライドします。

  ```
  PlsqlCGIEnvironmentList  SERVER_NAME  mylsnr1.mycompany.com
  PlsqlCGIEnvironmentList  SERVER_PORT  7777
  PlsqlCGIEnvironmentList  HOST mylsnr1.us.oracle.com:7777
  ```

いかなる場合でも、目的はアプリケーションに 2 番目のリスナーが存在しないかのように見せかけて、URL を生成することです。

3. 設定をテストし、すべての DAD に問題なくアクセスできることを確認します。

4. この設定では、mylsnr1.mycompany.com のメインのリスナーは、Oracle HTTP Server リスナーに対する合計の負荷に基づいて設定できます。mylsnr2.mycompany.com の 2 番目のリスナーは、実行される mod_plsql リクエストのみに基づいて細かくチューニングできます。

4.3.5 オーバーヘッドの問題

一部のストアド・プロシージャの実行時、mod_plsql に Describe のオーバーヘッドが発生することがあり、その結果、実行を正常終了するために、データベースへのラウンドトリップが 2 回余分に発生します。これは、パフォーマンスと密接な関係があります。

4.3.5.1 Describe のオーバーヘッド

PL/SQL プロシージャを実行するために、mod_plsql は、渡されるパラメータのデータ型を認識する必要があります。この情報に基づいて、mod_plsql は、各パラメータを配列またはスカラーとしてバインドします。プロシージャ・シグネチャを認識するための 1 つの方法は、プロシージャを実行前に記述することです。ただし、この方法は、各プロシージャを実行前に記述する必要があるため、効率的ではありません。Describe のオーバーヘッドを回避するには、mod_plsql でパラメータ名ごとに渡されるパラメータの数を調べるようにします。この情報を使用して、各変数のデータ型を推定します。このロジックは単に、渡される値が 1 つの場合はそのパラメータをスカラーとし、それ以外の場合は配列とします。これは、ほとんどの場合において機能しますが、1 つの値を配列パラメータに渡そうとしたり、複数の値をスカラーに渡そうとしている場合には機能しません。このような場合、プロシージャを実行する最初の試みに失敗します。mod_plsql は Describe コールを発行して、PL/SQL プロシージャのシグネチャを取得し、Describe 操作で取得した情報に基づいて各パラメータをバインドします。プロシージャは再実行され、結果が戻されます。

この Describe コールは、プロシージャに対して透過的に発生しますが、mod_plsql の内部では、ラウンドトリップが 2 回余分に発生します。1 回は実行に失敗したコールのため、もう 1 回は Describe コールのためです。

4.3.5.2 Describe のオーバーヘッドの回避

次のようにして、パフォーマンスの問題を回避できます。

- 柔軟なパラメータの受渡しを使用します。
- 配列に対して常に複数の値を渡すようにします。1 つの値の場合、プロシージャで無視されるダミー値を渡すことができます。
- 未使用の変数をデフォルトとする 2 パラメータ・スタイルのプロシージャを定義する、次の回避策を実行します。

  1. 元のプロシージャを内部的にコールする、プロシージャに相当するスカラーを定義します。たとえば、元のパッケージが次の例のようなものだとします。

     ```
     CREATE OR REPLACE PACKAGE testpkg AS
      TYPE myArrayType is TABLE of VARCHAR2(32767) INDEX BY binary_ integer;
      PROCEDURE arrayproc (arr myArrayType);
     END testpkg;
     /
     ```

  2. /pls/.../testpkg.arrayproc? arr= 1 のような URL コールを作成している場合、仕様を次のように変更します。

     ```
     CREATE OR REPLACE PACKAGE testpkg AS
      TYPE myArrayType is TABLE of VARCHAR2( 32767) INDEX BY binary_integer;
        PROCEDURE arrayproc (arr varchar2);
        PROCEDURE arrayproc (arr myArrayType);
     END testpkg;
     /
     ```

  3. プロシージャ arrayproc は、次のようになります。

     ```
     CREATE OR REPLACE PACKAGE BODY testpkg AS
     PROCEDURE arrayproc (arr varchar2) IS
       localArr myArrayType;
     BEGIN
       localArr( 1) := arr;
       arrayproc (localArr);
     END arrayproc;
     ```

4.3.6 柔軟なパラメータの受渡し（4 パラメータ）のオーバーヘッド

ラウンドトリップのオーバーヘッドは、PL/SQL プロシージャが古いスタイルの 4 パラメータ・インタフェースを使用している場合に発生します。PL/SQL Gateway は、最初に 2 パラメータ・インタフェースを使用してプロシージャの実行を試みます。これに失敗すると、PL/SQL Gateway は 4 パラメータ・インタフェースを使用します。つまり、すべての 4 パラメータ・インタフェースのプロシージャでは、ラウンドトリップが 1 回余分に発生します。

- 柔軟なパラメータの受渡しのオーバーヘッドの回避

  このオーバーヘッドを回避するには、対応するラッパーを作成して、2 パラメータ・インタフェースを使用しながら内部的に 4 パラメータ・インタフェースのプロシージャをコールするようにすることをお薦めします。もう 1 つのオプションは、元のプロシージャの仕様を変更して、渡されないパラメータをデフォルトとして 2 パラメータ・インタフェースに設定することです。4 パラメータ・インタフェースは、下位互換性のためにのみ提供されていますが、将来的には使用されなくなります。

- 柔軟なパラメータと感嘆符の使用

  Oracle HTTP Server の柔軟なパラメータの受渡しモードでは、PL/SQL プロシージャのプロシージャ名の前に感嘆符が付いていると想定しています。Oracle HTTP Server で使用されている自動検出方法はパフォーマンスと密接に関係するため、現在、Oracle HTTP Server での柔軟なパラメータの受渡しには感嘆符が必須です。Oracle HTTP Server では、各プロシージャは実行前に完全に記述されます。プロシージャ `Describe` コールにより、プロシージャのシグネチャが決まり、データベースへのラウンドトリップが必要となります。Oracle HTTP Server の PL/SQL Gateway では、エンド・ユーザーがプロシージャの前に感嘆符を追加して、明示的に柔軟なパラメータの受渡し規則を指定することにより、このラウンドトリップを回避します。

4.4 キャッシングのパフォーマンスを向上させるためのファイル・システム・キャッシュのチューニング

ファイル・システム・キャッシュを構成し、使用すると、OracleAS Portal アプリケーションおよび一般的な PL/SQL ベースの Web アプリケーションのパフォーマンスを向上させることができます。

この項の内容は、次のとおりです。

4.4.1 ファイル・システム・キャッシュのチューニングの概要

この項では、mod_plsql に関連するファイル・システム・キャッシュのチューニング・オプションについて説明します。キャッシュ・コンテンツは、オペレーティング・システム付属のファイル・システム・コールを使用してキャッシュされますが、キャッシュされたコンテンツは mod_plsql のメモリー領域には格納されません。mod_plsql のファイル・システム・キャッシュを使用していても、メモリー・ディスクなどの機能がオペレーティング・システムでサポートされ、それを使用するようにシステムが構成されている場合（一部の UNIX プラットフォームはメモリー・ディスク・ベースの高速格納をサポートしています）、キャッシュ・コンテンツはメモリーに存在することがあります。

この項の情報を使用すると、mod_plsql がファイル・システム・キャッシュを使用するように構成されている場合に、PL/SQL ベースの Web アプリケーションのパフォーマンスを向上させることができます。たとえば、OracleAS Portal ではファイル・システム・キャッシュを使用しています。そのため、ファイル・システム・キャッシュが適切にチューニングされれば、OracleAS Portal のパフォーマンスは向上します。

表 4-3 に、mod_plsql に対して設定できるキャッシュ関連のパラメータを示します。これらのパラメータを `cache.conf` ファイルに設定します。このファイルは、UNIX では *`ORACLE_HOME`*`/Apache/modplsql/conf` ディレクトリに、Windows では *`ORACLE_HOME`*`¥Apache¥modplsql¥conf` ディレクトリにあります。

注意： `conf` ディレクトリの `cache.README` ファイルには、各パラメータの詳細な説明およびパラメータ値の設定方法を示す例が含まれています。

表 4-3 mod_plsql cache.conf の構成パラメータの概要

パラメータ	説明
`PlsqlCacheCleanupTime`	キャッシュ・クリーン・アップ・ルーチンの実行間隔を設定します。 デフォルト：`Everyday 23:00`（現地時間の毎日午後 11 時にクリーン・アップ・ルーチンを実行） **関連項目：** 4.4.5 項「キャッシュ・クリーン・アップの構成」
`PlsqlCacheDirectory`	mod_plsql のキャッシュを保持するディレクトリを定義します。 デフォルト： UNIX システムでは、エラー・ログのデフォルト・ディレクトリは *`ORACLE_HOME`*`/Apache/modplsql/cache` です。 Windows システムでは、デフォルト・ディレクトリは *`ORACLE_HOME`*`¥Apache¥modplsql¥cache` です。 **関連項目：** 4.4.3 項「より高速なファイル・システムに存在するためのファイル・システム・キャッシュの構成」
`PlsqlCacheEnable`	ファイル・システム・キャッシュを使用できるようにします。 デフォルト：`On` **関連項目：** 4.4.2 項「ファイル・システム・キャッシュの有効化」

表 4-3 mod_plsql cache.conf の構成パラメータの概要（続き）

パラメータ	説明
PlsqlCacheMaxAge	キャッシュ・コンテンツのエージングを日単位で制御します。 デフォルト : 30（日） **関連項目 :** 4.4.4.2 項「PlsqlCacheMaxAge によるキャッシュのエージング日数の設定」
PlsqlCacheMaxSize	キャッシュに格納される個々のファイルの最大サイズをバイト単位で設定します。 デフォルト : 1048576（1MB） **関連項目 :** 4.4.4.3 項「PlsqlCacheMaxSize によるキャッシュ・ファイルの最大ファイル・サイズの設定」
PlsqlCacheTotalSize	キャッシュの合計サイズを制限します。この値はバイト単位で指定します。 デフォルト : 20971520（20MB） **関連項目 :** 4.4.4.1 項「PlsqlCacheTotalSize による合計キャッシュ・サイズの設定」

4.4.2 ファイル・システム・キャッシュの有効化

cache.conf の PlsqlCacheEnable パラメータにより、mod_plsql キャッシングが使用できるようになります。パフォーマンスを最大化するには、PlsqlCacheEnable パラメータの値を On に設定して有効にします。

注意 : PL/SQL キャッシングをサポートするアプリケーション（Oracle Portal など）でのみ、PlsqlCacheEnable を On に設定するメリットがあります。

4.4.3 より高速なファイル・システムに存在するためのファイル・システム・キャッシュの構成

この項では、別のディスク上にあるようにファイル・システム・キャッシュを構成する方法について説明します。ファイル・システム・キャッシュを使用し、より高速な別のディスク上にキャッシュを格納すると、OracleAS Portal や一般的な PL/SQL ベースの Web アプリケーションなど、ファイル・システム・キャッシュを使用するすべてのタイプの Web アプリケーションについて、パフォーマンスは向上します。

ファイル・システム・キャッシュを構成する際、キャッシュを別の物理ディスクまたはメモリー・ディスクのいずれかに設定できます。

ファイル・システム・キャッシュを別のディスク上に設定するには、次のようにします。

1. キャッシュのファイル・システムは次の場所にあるとします。

   UNIX の場合 : /u01/cache

   Windows の場合 : E:¥cache

2. 次のファイルを更新します。

   UNIX の場合 : *ORACLE_HOME*/Apache/modplsql/conf/cache.conf

   Windows の場合 : *ORACLE_HOME*¥Apache¥modplsql¥conf¥cache.conf

3. キャッシュ・パラメータの PlsqlCacheDirectory を次のように変更します。

   UNIX の場合 : PlsqlCacheDirectory /u01/cache

   Windows の場合 : PlsqlCacheDirectory E:¥cache

4.4.4 ファイル・システム・キャッシュのサイズ変更

この項の内容は、次のとおりです。

4.4.4.1 PlsqlCacheTotalSize による合計キャッシュ・サイズの設定

デフォルトのインストールでは、mod_plsql のファイル・システム・キャッシュ・サイズは 2097152 バイト（20MB）に設定されます。PL/SQL アプリケーションが OWA_CACHE パッケージを使用しない場合、またはパッケージを使用して少量のコンテンツをキャッシュする場合は、デフォルト設定で十分です。PL/SQL アプリケーションが大量のコンテンツを mod_plsql のファイル・システム・キャッシュにキャッシュする場合、より高い値の指定を検討する必要があります。

キャッシュ・サイズを制御するには、`cache.conf` ファイルに `PlsqlCacheTotalSize` パラメータを設定します。UNIX システムでは、このファイルは *ORACLE_HOME*`/Apache/modplsql/conf` ディレクトリにあります。Windows システムでは、このファイルは *ORACLE_HOME*`¥Apache¥modplsql¥conf` にあります。

高いキャッシュ・ヒット率を達成するのに十分なキャッシュ・サイズを設定する必要があります。頻繁にアクセスされるコンテンツがキャッシュされた状態を維持するのに十分な大きさのキャッシュ・サイズを設定するようにしてください。また、キャッシュ・サイズが大きくなりすぎないように、ディスク領域の量を制限することも重要です。キャッシュ・サイズを適切にチューニングすると、頻繁にアクセスされるコンテンツすべてを保持するために十分なキャッシュを提供する一方で、非常に大きなキャッシュの検索は非効率であることから、キャッシュ・サイズが大きくなりすぎるのを防ぐこともできます。

`PlsqlCacheTotalSize` の値はバイト数で指定します。1MB は 1048576 バイトです。この設定は、割り当てられるキャッシュ量に対する弱い制限です。場合によっては、次のクリーン・アップ操作までに、キャッシュ・サイズがこの制限を超えて大きくなることがあります。そのため、キャッシュ・サイズに対する強い制限は、使用する物理ハード・ディスク・サイズとなります。この制限に達すると、領域が使用可能になるまで、キャッシュ・コンテンツをディスクに書き出すことができません。

注意： `PlsqlCacheTotalSize` に使用できる最大値は、4294967296 バイト（4GB）です。

適度なキャッシュ・サイズを決定するには、次のようにします。

1. `httpd.conf` の `LogLevel` を `info` レベルに設定して mod_plsql パフォーマンス・ロギングをオンにし、mod_plsql ロギングを使用できるようにします。
2. 日次ベースで `error_log` を監視します。UNIX システムでは、エラー・ログのデフォルト・ディレクトリは *ORACLE_HOME*`/Apache/Apache/log` です。Windows システムでは、デフォルト・ディレクトリは *ORACLE_HOME*`¥Apache¥Apache¥log` です。

mod_plsql `error_log` エントリは、次のような形式です。

```
[info] mod_plsql: cachecleanup deleted=2571 max_age=96,2178852b  kept=1042,25585368b
time=128s limit=25600000b
```

意味は次のとおりです。

`deleted` は、クリーン・アップ・プロセスで削除されたキャッシュ・ファイルの数です。

`max_age` は、一定期間使用されていなかったために削除されたキャッシュ・ファイルの数および合計サイズです。

`kept` は、クリーン・アップ・プロセス後に保持されたキャッシュ・ファイルの数および合計サイズです。

time は、クリーン・アップの実行にかかった時間です。

limit は、合計キャッシュ・サイズです。これは、PlsqlCacheTotalSize 設定の値です。

エラー・ログのエントリは次のように解析します。

- 保持されたファイルの数と比較して多数のファイルが削除されている場合、これはキャッシュ・サイズが小さすぎることを明らかに示しています。おそらくキャッシュ・サイズを増やす必要があります。
- 保持されたファイルの数と比較して少数のファイルが削除されている場合、これはキャッシュ・サイズがおそらく大きすぎることを示しています。十分なディスク領域がある場合、そのままにしておくか、またはキャッシュ・サイズを減らしてディスク領域を解放することができます。

4.4.4.2 PlsqlCacheMaxAge によるキャッシュのエージング日数の設定

PlsqlCacheMaxAge パラメータを使用すると、キャッシュ・コンテンツの古さを制御できます。パラメータの値は日数単位で指定します。このパラメータのデフォルト値は 30（日）です。これは、キャッシュ・コンテンツが 30 日以内であれば、キャッシュに保存されることを意味します。30 日を過ぎると、コンテンツはクリーン・アップ・プロセス時に削除対象とみなされます。

mod_plsql error_log の max_age 情報には、キャッシュ・ファイルのエージング情報が示されます。サイトが非常に動的なサイトである場合、この設定をより小さい値に構成します。古いキャッシュ・コンテンツは通常、再利用されないため、より小さい値がキャッシュ・ヒット率に影響を及ぼすことがないためです。サイトに多くの静的ページが含まれる場合、PlsqlCacheMaxAge の値を増やし、クリーン・アップ・プロセスでキャッシュ・コンテンツが故意に削除されないようにします。

4.4.4.3 PlsqlCacheMaxSize によるキャッシュ・ファイルの最大ファイル・サイズの設定

PlsqlCacheMaxSize パラメータを使用すると、キャッシュに格納される個々のファイルの最大サイズを指定できます。このパラメータを使用することで、1 つのキャッシュ・ファイルでキャッシュ全体がいっぱいになる状態を回避できます。

このパラメータのデフォルト値は 1048576（バイト）です。一般に、このパラメータは、合計キャッシュ・サイズの約 1 ～ 3% の値に設定します。

注意： PlsqlCacheMaxSize に使用できる最大値は、4294967296 バイト（4GB）です。

4.4.5 キャッシュ・クリーン・アップの構成

キャッシュ・クリーン・アップ・パラメータにより、ファイル・システム・キャッシュを調査し、必要に応じてクリーン・アップする頻度が決まります。キャッシュ・クリーン・アップ・パラメータの PlsqlCacheCleanupTime は cache.conf ファイルに指定されています。頻度は、毎日、毎週、毎月に設定できます。毎週のクリーン・アップを指定する場合、曜日と時刻を指定できます。

mod_plsql の PlsqlCacheCleanupTime のデフォルト設定は、現地時間の毎日午後 11 時です。そのため、デフォルトでは、毎晩午後 11 時にクリーン・アップ・ルーチンが実行されます。毎月の頻度を選択した場合、クリーン・アップは毎月第 1 土曜日に行われます。

クリーン・アップの頻度が高すぎるとキャッシュ・ヒット率が低くなり、クリーン・アップの頻度が低すぎるとキャッシュのディスク使用量が過多になるため、このパラメータを適切に設定することが重要です。

クリーン・アップ・アクティビティは、mod_plsql error_log のエントリを使用して監視します。エントリを分析して、クリーン・アップ・パラメータの PlsqlCacheCleanupTime をチューニングします。

```
[info] mod_plsql: cachecleanup deleted=2571 max_age=96,2178852b  kept=1042,25585368b
time=128s limit=25600000b
```

次の事柄に注意してください。

- クリーン・アップ時間の値が大きい場合、クリーン・アップの頻度の設定が低すぎることを示している可能性があります。クリーン・アップ操作で、多数のキャッシュ・ファイルの調査または削除に時間がかかることをログが示している場合は、クリーン・アップの頻度を高くすることで、クリーン・アップ操作にかかる時間を減らすことができます。
- クリーン・アップ操作時に古さが理由で多数のファイルが削除されている場合、クリーン・アップの頻度が低すぎることを示しています。この場合、頻度を高くして、クリーン・アップで古いキャッシュ・コンテンツをより頻繁に削除できるようにします。

4.5 Oracle HTTP Server のディレクティブ

Oracle HTTP Server での PL/SQL のパフォーマンスを向上させるには、使用している構成に対して Oracle HTTP Server のディレクティブを適切にチューニングする必要があります。

関連資料 :

- 『Oracle Application Server パフォーマンス・ガイド』の Oracle HTTP Server のディレクティブの構成に関する項
- 『Oracle HTTP Server 管理者ガイド』の第 4 章「サーバー・プロセスの管理」および第 5 章「ネットワーク接続の管理」

A

よくある質問

- 特定のパターンを持つプロシージャ名へのアクセスを無効化するにはどうすればよいですか。
- ファイル ORACLE_HOME/Apache/Apache/conf/error_log にエラー HTTP-503 ORA-12154 があります。これは何を意味しますか。
- mod_plsql の起動時に、/DAD/package.procedure() または /DAD/package.procedure(123) という形式の URL が機能しないのはなぜですか。

mod_plsql とはなんですか。

mod_plsql は Oracle HTTP Server のプラグインであり、SQL*Net 接続を介してブラウザ・リクエストをデータベース・ストアド・プロシージャのコールにマッピングすることでデータベースと通信します。通常、仮想パスの /pls で示されます。mod_plsql ゲートウェイは、Web 上での PL/SQL ベースのアプリケーションの構築と運用をサポートします。PL/SQL ストアド・プロシージャはデータベース表からデータを取得し、Web ブラウザで表示する書式付きデータと HTML コードが含まれた HTTP レスポンスを生成できます。詳細は、『Oracle Database アプリケーション開発者ガイド - 基礎編』を参照してください。

PL/SQL Web Toolkit とはなんですか。

PL/SQL Web Toolkit を使用すると、Web アプリケーションを、Oracle データベース・サーバーに格納される PL/SQL プロシージャとして開発できます。このツールキット内のパッケージにより定義されるプロシージャ、ファンクションおよびデータ型を、独自のストアド・プロシージャで使用できます。詳細は、『Oracle Database アプリケーション開発者ガイド - 基礎編』を参照してください。

mod_plsql のリリースを確認するにはどうすればよいですか。

mod_plsql のリリースは、mod_plsql バイナリで oversioncheck スクリプトを実行すると判断できます。

UNIX プラットフォームでは、次のコマンドを発行します。

```
ORACLE_HOME/Apache/Apache/bin/oversioncheck ORACLE_HOME/Apache/modplsql/bin/modplsql.so
```

Windows プラットフォームでは、次のコマンドを発行します。

```
ORACLE_HOME¥Apache¥Apache¥bin¥oversioncheck ORACLE_HOME¥bin¥modplsql.dll
```

OWA パッケージのリリースを確認するにはどうすればよいですか。

1. SQL*Plus を使用し、任意のユーザーとしてデータベースに接続します。
2. 次のコマンドを実行します。

   ```
   select owa_util.get_version from dual;
   ```

   OWA パッケージのリリースが表示されます。たとえば、10.1.2.0.4 などです。

この問合せに失敗した場合は、OWA パッケージのリリースが古すぎるためにバージョニング機能がありません。新しいリリースにアップグレードすることをお薦めします。

OWA パッケージをインストールするにはどうすればよいですか。

詳細は、1.2 項「必須パッケージのインストール」を参照してください。

OWA パッケージをアンインストールするにはどうすればよいですか。

次のタスクを実行すると、OWA パッケージをアンインストールできます。

1. OWA パッケージのインストール元ディレクトリにナビゲートします。たとえば、次のようになります。

   ```
   cd ORACLE_HOME/Apache/modplsql
   ```

2. SQL*Plus を使用し、OWA パッケージの所有者として接続します。OWA パッケージの旧バージョンがある場合を除き、ここでは SYS ユーザーを使用する必要があります。
3. owadins.sql スクリプトを起動して OWA パッケージをアンインストールします。

データベースに重複してインストールされている OWA パッケージを検出してクリーン・アップするには、どうすればよいですか。

次の SQL 問合せを使用すると、OWA パッケージの位置を判断できます。

```
SELECT OWNER, OBJECT_TYPE
FROM   DBA_OBJECTS
WHERE  OBJECT_NAME = 'OWA'
```

次の結果が表示されます。

```
SQL>

1  SELECT OWNER, OBJECT_TYPE
2  FROM DBA_OBJECTS
3* WHERE OBJECT_NAME = 'OWA'

OWNER  OBJECT_TYPE
-----  -----------
SYS    PACKAGE
SYS    PACKAGE BODY
PUBLIC SYNONYM
```

この SQL 問合せよりも多数の行が表示される場合は、他のスキーマに古い OWA パッケージが存在し、それが mod_plsql ユーザーにとって問題になっている可能性があることを意味します。この場合は、すべてのバージョンの OWA パッケージをデータベースからアンインストールし、製品とともに出荷されている OWA パッケージを再インストールします。

mod_plsql を介して PL/SQL プロシージャにアクセス中に HTTP エラー・コードが戻されます。

mod_plsql では、Oracle HTTP Server ファイルの *ORACLE_HOME*/Apache/Apache/logs/error_log に詳細なエラー・メッセージが記録されます。このファイルをスキャンして問題を把握してください。mod_plsql のロギングの詳細は、「mod_plsql ではどのような種類のロギング機能を使用できますか。」を参照してください。

すべての独自 PL/SQL プロシージャで、Netscape では「ドキュメントにデータが含まれていません」というエラー、Internet Explorer では空白ページが戻されます。

データベースに OWA パッケージが重複してインストールされていると、この問題が発生する場合があります。詳細は、「データベースに重複してインストールされている OWA パッケージを検出してクリーン・アップするには、どうすればよいですか。」を参照してください。

パフォーマンスの高い PL/SQL プロシージャがありますが、mod_plsql を介した一部の HTTP リクエストに 15 秒以上かかることがあります。

この問題の最も一般的な原因は、中間層とバックエンド・データベースのキャラクタ・セットが一致せず、Oracle HTTP Server で `HTTP KeepAlive` が有効化されていることです。設定にこの種の誤りがあると、無効な Content-Length がブラウザに送信され、ブラウザは `KeepAliveTimeout` 間隔によってストリームがクローズされた場合にのみレスポンス・ストリームの終わりを検出します。この問題を解決するには、DAD とデータベースの `PlsqlNLSLanguage` パラメータが一致することを確認します。

これが問題の原因ではない場合、PL/SQL アプリケーションにパフォーマンス問題があると考えられます。これは次の方法で確認できます。

- mod_plsql のパフォーマンス・ロギングを有効にします。詳細は、「mod_plsql ではどのような種類のロギング機能を使用できますか。」を参照してください。
- パフォーマンスの問題がある URL を数回実行します。
- **dbProcTime** にレポートされた時間が `error_log` にないかスキャンします。この時間は、データベースから見たターゲット PL/SQL プロシージャにかかった時間を表します。この

値が高い場合は、標準のデータベース・プロファイリング方法を使用して、PL/SQL アプリケーションをデバッグする必要があります。

mod_plsql を使用して独自データベース上でアプリケーションを実行することはできますか。

はい。ただし、アプリケーションを実行する前に、データベースに OWA パッケージをインストールする必要があります。1.2 項「必須パッケージのインストール」を参照してください。

mod_plsql 用の DAD を作成するにはどうすればよいですか。

『Oracle HTTP Server 管理者ガイド』の「mod_plsql」を参照してください。

mod_plsql ではどのような認証モードを使用できますか。

第 2 章「mod_plsql を使用したアプリケーション・データベース・アクセスの保護」を参照してください。

mod_plsql クリーン・アップ・スレッドとはなんですか。

mod_plsql は、各 httpd プロセスでスレッドを起動します。このスレッドのジョブは、アイドル状態のデータベース・セッションとファイル・システムのキャッシュをクリーン・アップすることです。このスレッドを**クリーン・アップ・スレッド**と呼びます。

mod_plsql には、どのような種類のデータベース接続プーリング機能がありますか。

第 4 章「PL/SQL のパフォーマンスの最適化」を参照してください。

mod_plsql ではどのようにしてデータベース・セッションがクリーン・アップされますか。

mod_plsql では、`PlsqlIdleSessionCleanupInterval` の構成設定に基づいて、使用されていないデータベース・セッションがクリーン・アップされます。この他、プールされたデータベース・セッションから提供されるリクエストの数は、設定ディレクティブ `PlsqlMaxRequestsPerSession` により制御されます。HTTPD プロセスがシャットダウンされると、データベース・セッションがクローズされます。

プールされたデータベース接続が mod_plsql に存在するときに、データベースを再起動するとどうなりますか。

詳細は、3.11.3 項「接続プールでの停止中のデータベース接続の検出」を参照してください。

mod_plsql ではファイル・システム内でキャッシュされたコンテンツがどのようにクリーン・アップされますか。

クリーン・アップ・スレッドは、`PlsqlCacheCleanupTime` の設定に基づいてファイル・システム・キャッシュをスキャンします。デフォルトのクリーン・アップ時刻は、現地時間で毎日午後 11 時です。

URL に接頭辞 /pls を付けずに mod_plsql を起動できますか。

はい。mod_plsql は Oracle HTTP Server の `Location` ディレクティブを使用するため、mod_plsql で処理されるように任意の仮想パスを設定できます。

PL/SQL と mod_plsql のパフォーマンスを改善するにはどうすればよいですか。

第 4 章「PL/SQL のパフォーマンスの最適化」を参照してください。

mod_plsql ではどのような種類のロギング機能を使用できますか。

- mod_plsql では、デフォルトで Oracle HTTP Server `error_log` ファイルの *ORACLE_HOME*`/Apache/Apache/logs/error_log` にアラート、警告またはエラーが記録されます。mod_plsql で記録される情報の量は、Oracle HTTP Server の `httpd.conf` の `LogLevel` パラメータ設定で制御されます。デフォルトでは、このパラメータは `warn` に設定されます。
- また、次の手順に従って、mod_plsql のパフォーマンス・ロギングをリクエストごとに有効化することもできます。

  1. *ORACLE_HOME*`/Apache/Apache/conf/httpd.conf` を編集し、`LogLevel` を `info`（デフォルトは `warn`）に設定します。
  2. 次のコマンドを使用して Oracle HTTP Server を再起動します。

     ORACLE_HOME`/opmn/bin/opmnctl restartproc type=ohs`
  3. mod_plsql に対してなんらかの URL を発行し、ファイル *ORACLE_HOME*`/Apache/Apache/logs/error_log` が次のようなエントリで始まることを確認します。

     ```
     [Tue Apr 01 14:54:49 2003] [info] mod_plsql: [perf] 130.35.92.145
     /pls/app/htp.p status=200 user=scott reqTime=21ms connSU=(null),
     0ms connRO=(null),0ms connNSSO=HIT,1ms procTime=17ms sessionTidyTime=0ms
     cache=(null) cookie=(null),
     0ms pageCalls=0,0ms bytes=5 describe=No,0ms streamTime=0ms pid=175
     sessFile=(null) userFile=834¥0855 sysFile=470¥5949 cacheLevel=(null)
     cacheTime=0ms dbProcTime=15ms id=1049237685:130.35.92.145:373:1 spid=(null)
     qs=(null) requestTrace=(null) cookieLen=0 cookieValue=(null)
     reqUserTime=16ms assertUser=(null) subid=(null) authLevel=(null) oraError=0
     ```
- 最後に、mod_plsql でデバッグ・ロギングを有効化できます。これは最上位のロギング・レベルであり、アクティブ・サイトの場合はお薦めしません。

  注意：このロギング・モードは、オラクル社カスタマ・サポート・センターから要請された場合にのみ有効化してください。

  このモードでは、デバッグ・メッセージが Oracle HTTP Server の `error_log` ファイルに記録され、さらに mod_plsql 固有のログが *ORACLE_HOME*`/Apache/modplsql/logs` に作成されます。ログの位置は、*ORACLE_HOME*`/Apache/modplsql/conf/plsql.conf` 内で `PlsqlLogDirectory` ディレクティブを使用して設定可能です。デバッグ・レベルのロギングを有効化する手順は、次のとおりです。

  1. *ORACLE_HOME*`/Apache/modplsql/conf/plsql.conf` を編集し、`PlsqlLogEnable` を **On**（デフォルトは **Off**）に設定します。
  2. 次のコマンドを使用して Oracle HTTP Server を再起動します。

     ORACLE_HOME`/opmn/bin/opmnctl restartproc type=ohs`

mod_plsql には高可用性についてどのような考慮事項がありますか。

高可用性のために、mod_plsql ベースのアプリケーションでは次のことを認識する必要があります。

- mod_plsql の構成パラメータ `PlsqlDatabaseConnectString` には、Oracle Internet Directory の LDAP 検索を介して名前解決が行われるように、接続文字列形式の `NetServiceNameFormat` を使用する必要があります。これにより、データベースの `host:port:service_name` 情報を中央のリポジトリで設定でき、必要に応じた RAC ノードの追加または削除が容易になります。
- mod_plsql では、停止中のデータベース接続が自動的に検出されます。詳細は、3.11.3 項「接続プールでの停止中のデータベース接続の検出」を参照してください。

データベースがファイアウォールで分離されている場合、mod_plsql ではどのような考慮事項がありますか。

mod_plsql を実行する中間層とバックエンド・データベースの間にファイアウォールが存在する場合は、mod_plsql でアイドル・セッションのクリーン・アップ間隔を、ファイアウォールでの間隔よりも短く設定する必要があります。これにより、mod_plsql が確立した接続をファイアウォールがクローズすることはなくなります。

注意： mod_plsql のアイドル・セッション・クリーン・アップ間隔は、*ORACLE_HOME*/Apache/modplsql/conf/plsql.conf 内のパラメータ PlsqlIdleSessionCleanupInterval を使用して設定できます。デフォルト値は 15 分です。

PL/SQL アプリケーションに対して異なるホスト名、ポートまたは request_protocol をアサートするには、どうすればよいですか。

- Oracle HTTP Server インスタンスの手前に Web Cache またはロード・バランシング・ルーターが存在する場合は、サイトのホスト名とポートを Web Cache または LBR のホスト名およびポートとしてアサートする必要があります。このような場合は、Oracle HTTP Server 設定ディレクティブ ServerName および Port を使用してアサートすることをお薦めします。なんらかの理由でホスト名とポートを Oracle HTTP Server レベルでアサートしない場合は、mod_plsql 設定ディレクティブ PlsqlCGIEnvironmentList を使用して、mod_plsql で実行中の PL/SQL アプリケーションに対してのみ異なるホスト名とポートをアサートできます。たとえば、次のようになります。

  - PlsqlCGIEnvironmentList SERVER_NAME=lbr.us.oracle.com

    かわりに Oracle HTTP Server の httpd.conf 内の ServerName ディレクティブを使用することを考慮してください。

  - PlsqlCGIEnvironmentList SERVER_PORT=9999

    かわりに Oracle HTTP Server の httpd.conf 内の Port ディレクティブを使用することを考慮してください。

  - PlsqlCGIEnvironmentList
    HTTP_HOST=myservername.us.oracle.com:9999

    SERVER_NAME:SERVER_PORT の組合せです。

- 同様に、サイトが外部からは SSL としてアクセスされるが、内部的には非 SSL モードで（間に SSL アクセラレータを使用して）稼働する場合は、PL/SQL アプリケーションで非 SSL リンクのかわりに SSL リンクを生成できるように、REQUEST_PROTOCOL を HTTPS としてアサートできます。たとえば、次のようになります。

  PlsqlCGIEnvironmentList REQUEST_PROTOCOL=https

特定のパターンを持つプロシージャ名へのアクセスを無効化するにはどうすればよいですか。

詳細は、2.1.2 項「mod_plsql での PlsqlExclusionList ディレクティブへのルールの追加」を参照してください。

ファイル ORACLE_HOME/Apache/Apache/conf/error_log にエラー HTTP-503 ORA-12154 があります。これは何を意味しますか。

このエラーは、mod_plsql からデータベースに接続できないことを意味します。

次のことを確認してください。

1. データベースが稼働中であること。
2. DAD に指定されているユーザー名およびパスワード情報が正しいこと。
3. 中間層が、DAD の `PlsqlDatabaseConnectString` パラメータを使用してデータベースに接続できること。

ほとんどの場合、この問題が発生するのは、SQL*Net が *`ORACLE_HOME`*`/network/admin` にある設定情報を使用して接続文字列パラメータを解決できないためです。

- `TNSFormat` または `NetServiceNameFormat` を使用して設定したエントリの場合は、`tnsping dad_connect_string` を使用して接続文字列情報を確認します。たとえば、次のようになります。

  ```
  tnsping "cn=iasdb,cn=oraclecontext"
  ```

  または

  ```
  tnsping iasdb.us.oracle.com
  ```

- `SIDFormat` および `ServiceNameFormat` を使用して設定したエントリの場合は、データベース・リスナーの hostname、port および SID/service_name 情報が一致していることを確認します。その後、SQL*Plus で DAD にあるユーザー名、パスワードおよび接続文字列を使用してデータベースに接続できることを確認します。

接続できない場合、トラブルシューティング情報については Oracle SQL*Net のマニュアルを参照してください。

注意： DAD のデフォルトの接続文字列パラメータは、Oracle Internet Directory で LDAP 検索を介して解決されるように設定されます。`ldap.ora` を変更した場合は、その変更に mod_plsql からアクセスできるように Oracle HTTP Server を再起動する必要があります。

---

mod_plsql の起動時に、/DAD/package.procedure() または /DAD/package.procedure(123) という形式の URL が機能しないのはなぜですか。

mod_plsql では次の形式の URL をサポートしていません。

- `/DAD/package.procedure()`
- `/DAD/package.procedure(123)`

たとえば、次のようになります。

`http://www.acme.com:9000/pls/mydad/mypackage.myproc()` または
`http://www.acme.com:9000/pls/mydad/mypackage.myproc(123)`

サポートされている URL の形式の詳細は、3.3 項「mod_plsql の実行」を参照してください。

索引

記号

数字

B

C

D

E

G

H

K

L

M

O

P

ふ

ゆ

ろ