Oracle
DBA & Developer Days 2011

日本オラクル、今年最大の技術トレーニングイベント

2011年11月9日(水)～11月11日(金) シェラトン都ホテル東京

ORACLE

PL/SQL上級テクニック

日本オラクル株式会社　製品事業統括
プリンシパルエンジニア　中家 裕之

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行
  ～PL/SQLでMapReduce?～
- Appendix

Agenda

- あまり語られないUTL_FILEの仕様
  - UTL_FILEの基本
    - UTL_FILEとセキュリティ
    - その他の仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix

UTL_FILEとは

- UTL_FILE ： PL/SQLでファイル操作を行うための
  ユーティリティ・パッケージ

バージョン	拡張・追加された機能
8.0.5	1行あたり32,767バイトまでのI/Oに対応
9.1.0	各国語キャラクタセットを使ったファイルのI/O
9.2.0	ディレクトリ・オブジェクトへの対応
	ファイルの複製・削除・改名・属性取得
10.1.0	バイナリファイルのI/O

UTL_FILE中のFUNCTION/PROCEDURE(1/2)

名称	F/P	拡張・追加された機能
FCLOSE	P	ファイルをクローズ
FCLOSE_ALL	P	オープンしているファイル・ハンドルをすべてクローズ
FCOPY	P	ファイルをコピー
FFLUSH	P	バッファ中データの強制書き込み
FGETATTR	P	ファイルの属性を取得
FGETPOS	F	ファイル内の現行の相対オフセット位置をバイト数で戻す
FOPEN	F	ファイルオープン
FOPEN_NCHAR	F	Unicodeファイルをオープン
FREMOVE	P	ファイルを削除
FRENAME	P	ファイル名を変更、ディレクトリ変更も可
FSEEK	P	ファイルをシーク
GET_LINE	F	ファイルから1行読み込む

UTL_FILE中のFUNCTION/PROCEDURE(2/2)

名称	F/P	拡張・追加された機能
GET_LINE_NCHAR	F	ファイルから1行Unicodeで読み込む
GET_RAW	F	ファイルをバイナリデータとして読む
IS_OPEN	F	ファイルがオープンされているかどうかを返す
NEW_LINE	P	ファイルに改行コードを書き込む
PUT	P	ファイルに文字列を書き込む
PUT_LINE	P	ファイルに文字列を改行コード付きで書き込む
PUT_LINE_NCHAR	P	ファイルにUnicodeで文字列を改行コード付きで書き込む
PUT_NCHAR	P	ファイルにUnicodeで文字列を書き込む
PUTF	P	ファイルに文字列を書式付きで書き込む
PUTF_NCHAR	P	ファイルに文字列をUnicodeで書式付きで書き込む
PUT_RAW	P	ファイルにバイナリデータを書き込む

UTL_FILE読み込みサンプル

```
DECLARE
   v_filedir varchar2(200) := 'DMP_DIR';
   v_filename varchar2(100) := 'alert_orcl.log';
   v_filehandle UTL_FILE.FILE_TYPE;
   v_line varchar2(255);
   v_linesize number := 32767;
BEGIN
  v_filehandle := UTL_FILE.FOPEN(v_filedir,v_filename,'R',v_linesize);
  LOOP
    UTL_FILE.GET_LINE(v_filehandle,v_line);
    IF v_line like 'ORA%' THEN
     DBMS_OUTPUT.PUT_LINE(v_line);
    END IF;
  END LOOP;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('処理が終了しました。');
    UTL_FILE.FCLOSE(v_filehandle);
END;
/
```

ファイルのハンドル用のデータ型

ファイルをオープン
- 第一引数:ディレクトリ(・オブジェクト)名
- 第二引数:ファイルハンドル
- 第三引数:オープンモード
  - 'R':読み取り
- 第四引数:1行のサイズ
  (デフォルトは1024バイト)

ファイルを1行読み込み
- 第一引数:ファイルハンドル
- 第二引数:読み取ったデータを格納する変数

ファイルの最後に達すると
NO_DATA_FOUND例外が発生

ファイルをクローズ
- 第一引数:ファイルハンドル

UTL_FILE書き出しサンプル

```
declare
  v_deptno char(2) := '10';
  v_filedir varchar2(200) :='DMP_DIR';
  v_filename varchar2(50) := 'utlfiles.txt';
  v_filehandle UTL_FILE.FILE_TYPE;
  CURSOR emp_info IS select empno,ename,sal from emp where deptno=v_deptno;
begin
  v_filehandle := UTL_FILE.FOPEN(v_filedir,v_filename,'w');
  UTL_FILE.PUTF(v_filehandle,'Created: %s¥n',SYSDATE);
  UTL_FILE.NEW_LINE(v_filehandle);
  UTL_FILE.PUT_LINE(v_filehandle,'DEPTNO: ' || v_deptno);
  FOR emp_info_record IN emp_info LOOP
    UTL_FILE.PUTF(v_filehandle,' %s: %s: %s¥n',
      emp_info_record.empno,
      RPAD(emp_info_record.ename,30),
      LPAD(to_char(emp_info_record.sal,'$99,999.00'),12,' '));
  END LOOP;
  UTL_FILE.FCLOSE(v_filehandle);
END;
/
```

第三引数:オープンモード
- 'W':新規・上書き書き込み
- 'A':追記

改行コードを出力

ファイルを1行書き込み(改行コード自動付加)
- 第一引数:ファイルハンドル
- 第二引数:書き込みデータ

ファイルを指定書式でフォーマットして書き込み
(改行コードは自前で付加)
- 第一引数:ファイルハンドル
- 第二引数:書き込みデータ
- 第三引数以降:書き込み対象変数。最大5つまで

Agenda

- あまり語られないUTL_FILEの仕様
  - UTL_FILEの基本
  - UTL_FILEとセキュリティ
  - その他の仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix

UTL_FILEとファイルの権限(Linux/UNIX)

- サーバープロセスのOSユーザーがownerになる
  - listener経由か否かでサーバープロセスのOSユーザーが決まる

接続	OSユーザー
ローカル接続	クライアントのプロセスを起動したユーザー
listener経由接続	listenerを起動したユーザー
共有サーバー接続	インスタンスを起動したユーザー

  - パーミッションはプロセス起動時に決まる
    →umaskを変更したい際は起動前に変更する必要がある
  - いかなるumaskであっても、一般ユーザーの書き込み権限はセキュリティのために付与されない

UTL_FILEとファイルの権限(Windows)

- Administratorsグループの権限でのアクセスとなる
- UNC指定による他のサーバーのディスクへのI/Oは、Administrator権限を持つアカウントでインスタンスを起動する必要あり
  - デフォルトのSYSTEMアカウントはリモートアクセスができない
  - サービス「OracleService???」(???はインスタンス名)の「プロパティ」画面の「ログオン」タブにて起動ユーザーの確認と変更が可能
  - 10g以降のRAC環境では加えてOCRに該当アカウントの追加が必要
    - 手順はAppendixに記載

UTL_FILEでのディレクトリ指定:概要

- セキュリティを保つため、UTL_FILEでアクセス対象となるディレクトリには、事前にアクセス許可を設定する必要がある

方法1	• 初期化パラメータUTL_FILE_DIRにアクセス対象ディレクトリを列挙
方法2	• ディレクトリ・オブジェクトを作成し、ディレクトリ・オブジェクトにI/Oの権限を付与

- サブディレクトリへのアクセスは不可
- シンボリックリンク、ハードリンクのファイルアクセス不可
- シンボリックリンクを含むパスの指定不可

UTL_FILEでのディレクトリ指定:UTL_FILE_DIR

- 初期化パラメータUTL_FILE_DIRにアクセス対象ディレクトリを列挙
  - 「*」を指定すると、アクセス権のある全ディレクトリにアクセス可能
    →セキュリティ面で本指定は非推奨
  - 本パラメータを反映させるためにはインスタンスの再起動が必要

```
SQL> alter system set utl_file_dir = '/tmp/d1','/tmp/d2' scope=spfile;
```

UTL_FILEでのディレクトリ指定: ディレクトリ・オブジェクト(1)

- ディレクトリ・オブジェクト: アクセス対象のディレクトリ(フォルダ)を定義したオブジェクト
  - このオブジェクトにI/Oの権限を付与して初めてアクセス可能
  - オブジェクトを介することで以下を実現
    - ファイルI/Oに対するセキュリティを強化
    - ディレクトリの仮想化による、コーディング内でのディレクトリ指定の不要化

UTL_FILEでのディレクトリ指定: ディレクトリ・オブジェクト(2)

- ディレクトリ・オブジェクトの指定方法
  1. ディレクトリ・オブジェクトを作成
     - SQL> CREATE DIRECTORY ディレクトリ名称 as ' 実ディレクトリ名 ';
     - CREATE DIRECTORY権限が必要
  2. ディレクトリ・オブジェクトの利用権限を付与
     - 読み取り権限
       SQL> GRANT READ ON DIRECTORY *ディレクトリ名称* TO *ユーザー/ロール*;
     - 書き込み権限
       SQL> GRANT WRITE ON DIRECTORY ディレクトリ名称 TO ユーザー/ロール;
     - 両方まとめて指定可能
       SQL> GRANT READ, WRITE ON DIRECTORY ディレクトリ名称 TO ユーザー/ロール;

UTL_FILEでのディレクトリ指定:
UTL_FILE_DIRとディレクトリ・オブジェクトの使い分け

- 基本的にディレクトリ・オブジェクトの利用を推奨
  - ディレクトリ・オブジェクトの方がセキュリティ観点で優れている
  - I/O先ディレクトリ(フォルダ)パス名を変更の際にコーディングの修正が不要
  - アクセス対象ディレクトリの追加変更削除の際にインスタンス再起動が不要
- UTL_FILE_DIRを使うケースはUNC指定を行う場合のみ
  - ディレクトリ・オブジェクトでUNC指定は未サポート

Agenda

- あまり語られないUTL_FILEの仕様
  - UTL_FILEの基本
  - UTL_FILEとセキュリティ
  - その他の仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix

UTL_FILEと排他制御

- UTL_FILE_DIRにファイル同時アクセスの排他制御を行う機能はない
  - 複数のアプリが同時に書きこむ可能性がある場合は、UTL_FILE.IS_OPEN関数などで事前チェックの上オープンすること

UTL_FILEと文字コード

- ファイルアクセスの際の文字コードはDBのキャラクタセット
  - NLS_LANGによる文字コード変換は無効
  - DBのキャラクタセットと異なるファイルを読み書きする際は、CONVERT関数を使ってコード変換処理を追加する
    - バイナリファイルとしてのI/Oが必要
- UnicodeのBOMの対応
  - BOM(Byte Order Mark):16bit単位のエンコーディングにてエンディアンを規定するデータ

キャラクタセット	Read	Write
Unicode系	BOMを読み飛ばす	BOMを書かない
非Unicode系	BOMを読む	BOMを書かない

  - BOMを出力したい場合はバイナリファイルとしてのI/Oが必要

Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
  - コンパイル時の最適化
    - モジュールのメモリ常駐化
    - ネイティブコンパイル
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix

コンパイル時の最適化

- 初期化パラメータPLSQL_OPTIMIZE_LEVELにて調整
  - Oracle10g以降で指定可能

値	最適化の動き	実行速度	コンパイル速度	ソースの問題の追及のしやすさ
0	•9iまでの動きと同じで最適化を行わない •遅くなる可能性あり	遅	速	易
1	•軽度の最適化を行う •ソースコードの順番の入れ替えは発生しない			
2	•デフォルト値 •中度の最適化を行う •ソースコードの順番の入れ替えが発生する			
3	•11gのみに存在 •大幅な最適化を行う •ソースコードの順番の入れ替えが発生する •ソースコードの置き換えや追加が発生する可能性がある	速	遅	難

Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
  - コンパイル時の最適化
  - モジュールのメモリ常駐化
  - ネイティブコンパイル
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix

モジュールのメモリ常駐化

- メモリ常駐化(PIN)により解析フェーズをスキップし、起動速度を向上させることができる
- DBMS_SHARED_POOLパッケージを使用
  - KEEPプロシージャを使用して常駐させる
    - SQL> exec DBMS_SHARED_POOL.KEEP('モジュール名', 'フラグ')
    - フラグは以下の3種類
      - P:パッケージ、プロシージャ、ファンクション
      - R:トリガー(R7.3以降)
      - Q:順序(R7.3以降)
  - 常駐を解除する場合はUNKEEPプロシージャを使用
    - SQL> exec DBMS_SHARED_POOL.UNKEEP('モジュール名', 'フラグ')
  - CREATE DATABASE文でDBを作成した場合は、dbmspool.sqlをSYSユーザーで実行してDBMS_SHARED_POOLパッケージを作成する必要あり
  - 初期状態ではSYSユーザー以外に実行権限が付与されていない

モジュールのメモリ常駐化実行例

```
SQL> exec dbms_shared_pool.keep('scott.proc1', 'p')

PL/SQLプロシージャが正常に完了しました。

SQL> set serveroutput on

SQL> exec dbms_shared_pool.sizes(0)
SIZE(K) KEPT    NAME
------- ------  ----------------------------------------------------------
635             SYS.STANDARD                        (PACKAGE)
(中略)
25      YES     SCOTT.PROC1                         (PROCEDURE)
(中略)
```

大文字・小文字は問わない

DBMS_SHARED_POOL.SIZESを実行する前に必要

引数のサイズ以上の、共有プール上のモジュール、SQLを表示

KEPT列が「YES」であれば常駐している

モジュールのメモリ常駐に関するTips

- 優先して常駐させるもの
  - 利用頻度の高いもの
  - Oracleのユーティリティ・パッケージ
    - DBMS_OUTPUT, DBMS_SQL, DBMS_SYS_SQL, DBMS_UTILITY, STANDARD, UTL_FILE他、システムで多用しているパッケージを優先して常駐
    - DBMS_STANDARDは常駐不可
  - 実行速度が非常に重視されるもの
- インスタンス起動直後の常駐を推奨
  - メモリの断片化の発生を防げる
  - STARTUPのシステム・イベント・トリガーを利用すると簡単(8i以降)

インスタンス起動時のメモリ常駐サンプル

```
CREATE OR REPLACE TRIGGER db_start_pin AFTER STARTUP ON DATABASE
BEGIN
  sys.dbms_shared_pool.keep('SYS.STANDARD', 'P');
  sys.dbms_shared_pool.keep('SCOTT.PROC1', 'P');
  sys.dbms_shared_pool.keep('SCOTT.TRG_INS_EMP', 'T');
  sys.dbms_shared_pool.keep('SCOTT.SEQ1', 'Q');
END;
/
```

インスタンス起動時に起動させるトリガー作成の際に指定

Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
  - コンパイル時の最適化
  - モジュールのメモリ常駐化
  - ネイティブコンパイル
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix

ネイティブコンパイルの概要

- デフォルトの中間モジュールではなく、ネイティブモジュールにコンパイル
  →C言語と同等の処理速度に
  - Oracle9i以降の機能
  - SQL部分は変換対象外
    →処理時間のうち、非SQL部分が占める割合が多いほど高速化の恩恵が高い
  - 無名PL/SQLブロックは変換対象外
  - コンパイル方法や、PL/SQL, SQLからの呼び出し方に変更は入らない
  - PL/SQLのユーティリティ・パッケージのネイティブコンパイルも可能
  - デバッグなど一部のシステム系機能が利用できなくなる
    →モジュール完成後のネイティブコンパイルを推奨
  - 消費メモリは非ネイティブコンパイル時より増える傾向にある

ネイティブコンパイル機能の比較

バージョンを経る度に実施方法が簡素化

バージョン	共有ライブラリ (so/dll)	コンパイラ	Windows版
9i	•作成される •ライブラリ格納ディレクトリを作成する必要がある	必要	非サポート
10g	•作成される •ライブラリ格納ディレクトリを作成する必要がある	必要	サポート
11g	•作成されない •ライブラリ格納ディレクトリを作成する必要がない	不要	サポート

ネイティブコンパイルの実施方法(DB11g)

- 初期化パラメータPLSQL_CODE_TYPEの設定のみ
  - ALTER SESSION/SYSTEM文での変更も可能

値	動作
INTERPRETED	•デフォルト •PL/SQLバイトコード形式にコンパイル
NATIVE	•システム固有のコード(ネイティブコード)形式にコンパイル

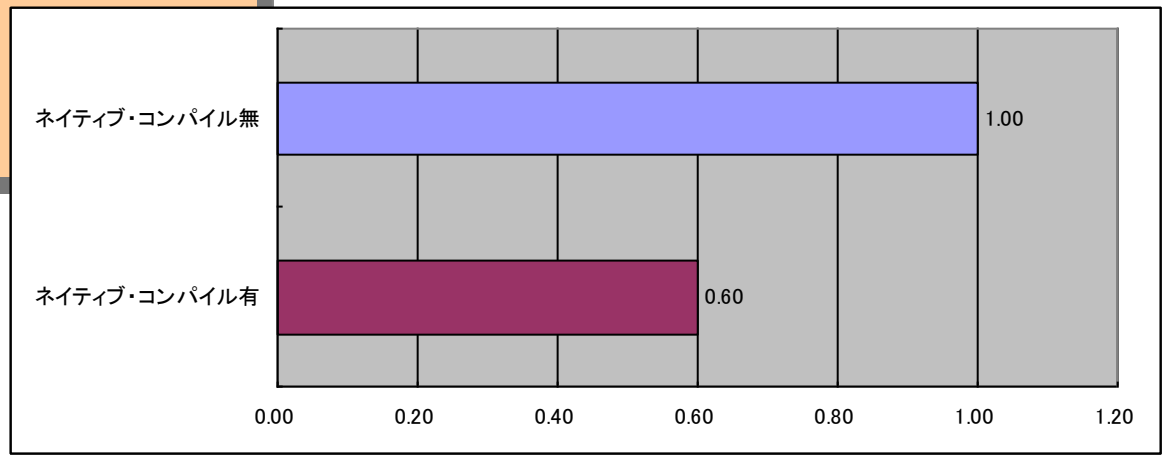
- モジュールごとのコンパイル・タイプはALL_PLSQL_OBJECT_SETTINGSビューのPLSQL_CODE_TYPE列で調査可能

ネイティブコンパイルの実行結果

```
CREATE OR REPLACE PROCEDURE test1 IS
  TYPE test_tbl_type IS TABLE OF pls_integer
    INDEX BY pls_integer;
  test_tbl test_tbl_type;
  PROCEDURE test (
    arg_cnt IN pls_integer,
    arg_tbl IN OUT NOCOPY test_tbl_type
  )IS
  BEGIN
    FOR i IN arg_tbl.FIRST .. arg_tbl.LAST
    LOOP
      arg_tbl(i) := arg_tbl(i) + i;
    END LOOP;
  END;
-- 隣へ続く
```

```
BEGIN
  FOR cnt in 0 .. 10000 LOOP
    test_tbl(cnt) := cnt;
  END LOOP;
  FOR cnt in 0 .. 10000 LOOP
    test(cnt, test_tbl);
  END LOOP;
END;
/
```

```
exec test1;
```

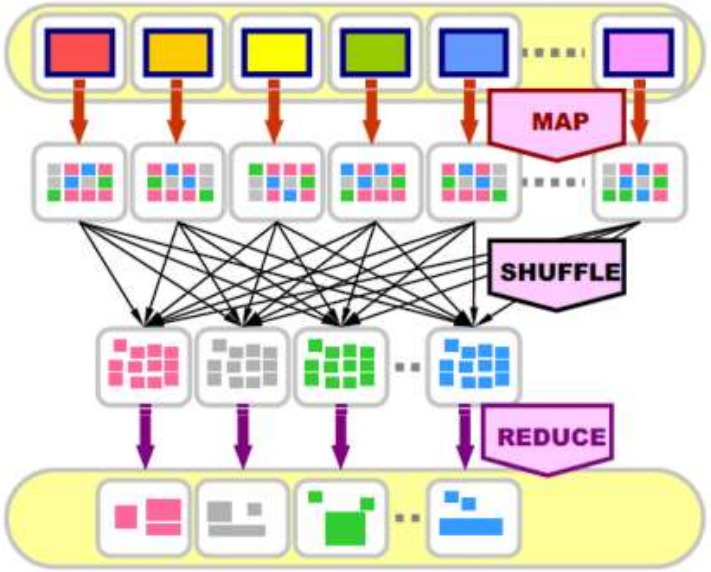


Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
  - MapReduceとPL/SQL
    - 表関数の概要
    - パラレルパイプライン表関数の解説
- Appendix

MapReduceとは?

- 分散並列処理のひとつ
  - Map処理とReduce処理のみ実装すれば、他の処理はMapReduce基盤ソフトが実施
    - Map:入力のKey-Valueデータを異なるKey-Valueデータに変換(分散)
    - Reduce:同一キーのKey-Valueを出力形式に変換(集約)
  - Shuffle:Mapの出力を同一キーにまとめ、Reduceに渡す
  - 代表的な実装はHadoop
- Oracle DatabaseとHadoopの連携
  →Appendix参照



図の出典:平成21年度産学連携ソフトウェア工学実践事業(高信頼クラウド実現用ソフトウェア開発(分散制御処理技術等に係るデータセンターの高信頼化に向けた実証事業))事業成果報告書 P1-3
http://www.meti.go.jp/policy/mono_info_service/joho/downloadfiles/2010software_research/clou_dist_software.pdf

PL/SQLでのMapReduceの考え方

- 分散並列処理としてはパラレルクエリー、パラレルDMLで同等の処理が実現可能
  - Hadoopより遥かに簡単
    - MapReduce v.s. SQL
- PL/SQLロジックの分散並列処理は？

パラレルパイプライン表関数で対応可

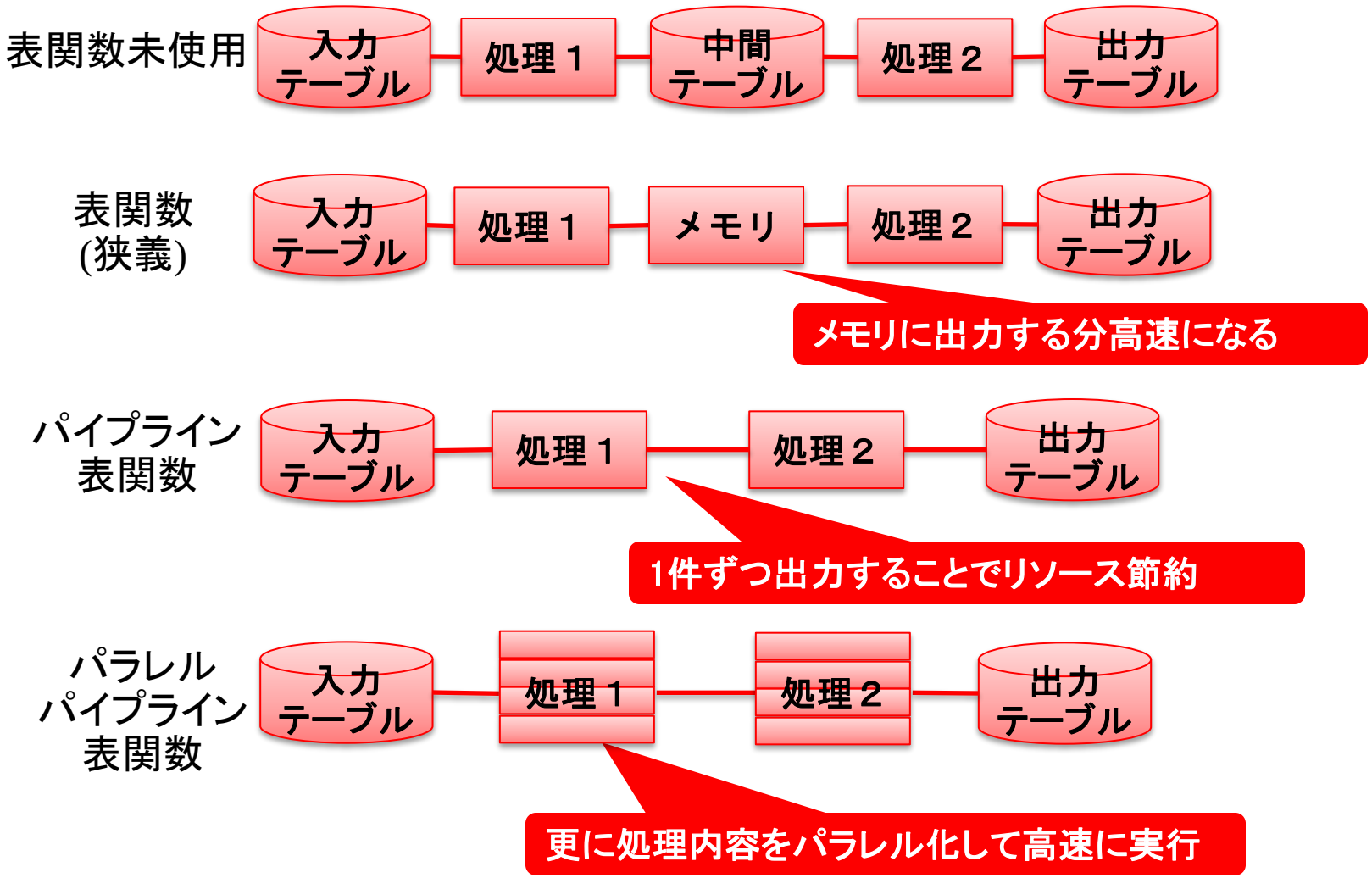
Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
  - MapReduceとPL/SQL
  - 表関数の概要
  - パラレルパイプライン表関数の解説
- Appendix

表関数とは

- 戻り値がテーブル(ネスト表)のFunction
- TABLE関数を経由させることで、読み取り専用テーブルとしてアクセスできる
  - SQL> SELECT * FROM TABLE(table_function1);
  - 引数がネスト表であれば表関数のネストが可能
    - SQL> SELECT * FROM
      TABLE(table_function2(TABLE(table_function1)));
- Oracle9i以降の機能

表関数の種類と動き



表関数作成前準備

- 戻り値となるネスト表のオブジェクトを定義

```
create or replace type emp_type as object
(empno number(5), ename varchar2(10));
/

create or replace type emp_tbl_type as table of emp_type;
/
```

オブジェクト型を定義する際のキーワード

TYPEはストアドの一種なので、「/」を入力しないと実行されない

ネスト表のオブジェクトを定義する際のキーワード

表関数サンプル

```
create or replace function get_emp_tbl(arg1 number)
  return emp_tbl_type is
    wk_emp_tbl emp_tbl_type;
begin
  select emp_type(empno, ename) bulk collect into wk_emp_tbl
    from emp where deptno = arg1;
  return wk_emp_tbl;
end;
/
```

戻り値としてネスト表を指定

コンストラクタ関数を通じてオブジェクト型のデータを生成

コンストラクタ関数:
オブジェクト型を作成すると自動で作成される、オブジェクト型と同じ名前の、オブジェクトの初期生成を行う関数。引数はオブジェクトを構成する一連のデータとなる。

表関数実行例

- 戻り値となるネスト表のオブジェクトを定義

```
SQL> select get_emp_tbl(10) from dual;

GET_EMP_TBL(10)(EMPNO, ENAME)
--------------------------------------------------------------------------------
EMP_TBL_TYPE(EMP_TYPE(7782, 'CLARK'), EMP_TYPE(7839, 'KING'), EMP_TYPE(7934, 'MILLER'))

SQL> select * from table(get_emp_tbl(10));

     EMPNO ENAME
---------- ----------
      7782 CLARK
      7839 KING
      7934 MILLER
```

SQL*Plusではオブジェクト型のデータはコンストラクタ関数を使用した形式で表示される

TABLE関数を通すと通常のテーブルと同様の検索結果表示になる

パイプライン表関数サンプル

```
create or replace function get_emp_tbl_p(arg1 number)
  return emp_tbl_type pipelined is
begin
  for rec in (select empno, ename from emp where deptno = arg1) loop
    pipe row (emp_type(rec.empno, rec.ename));
  end loop;
  return;
end;
/
```

PIPELINED句を付与する

PIPE ROW文にて、1レコードずつ出力する

RETURN文に出力変数は指定しない

Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
  - MapReduceとPL/SQL
  - 表関数の概要
  - パラレルパイプライン表関数の解説
- Appendix

パラレルパイプライン表関数サンプル

パッケージ定義部

```
create or replace package map_and_reduce as
  type clob_rec is record (data clob);
  type key_value_type is record (key char(1 char), value number);
  type key_value_type_tbl is table of key_value_type;
  type ref_cur_clob is ref cursor return clob_rec;
  type ref_cur_kv is ref cursor return key_value_type;
  function map (p_cur in ref_cur_clob)
    return key_value_type_tbl pipelined
    parallel_enable(partition p_cur by any);
  function reduce (p_cur in ref_cur_kv)
    return key_value_type_tbl pipelined
    cluster p_cur by (key)
    parallel_enable(partition p_cur by hash(key));
end map_and_reduce;
/
```

パラレルパイプライン表関数サンプル

パッケージ本体(map部分)

```
create or replace package body map_and_reduce as
  key_value key_value_type;

function map (p_cur in ref_cur_clob)
  return key_value_type_tbl pipelined
  parallel_enable(partition p_cur by any) is
    rec clob_rec;
begin
  loop
    fetch p_cur into rec;
    exit when p_cur%notfound;
    for i in 1 .. length(rec.data) loop
      key_value.key := substr(rec.data, i, 1);
      key_value.value := 1;
      pipe row(key_value);
    end loop;
  end loop;
  return;
end map;

(続く)
```

パラレルパイプライン表関数サンプル

パッケージ本体(reduce部分前半)

```
function reduce (p_cur in ref_cur_kv)
  return key_value_type_tbl pipelined
  cluster p_cur by (key)
  parallel_enable(partition p_cur by hash(key)) is
    rec_value key_value_type;
    type count_tbl_type is table of pls_integer index by varchar2(1 char);
    count_tbl count_tbl_type;
    i varchar2 (1 char);
begin
  loop
    fetch p_cur into rec_value;
    exit when p_cur%notfound;
    if count_tbl.exists(rec_value.key) then
      count_tbl(rec_value.key) := count_tbl(rec_value.key) + rec_value.value;
    else
      count_tbl(rec_value.key) := 1;
    end if;
  end loop;

(続く)
```

パラレルパイプライン表関数サンプル

パッケージ本体(reduce部分後半)

```
i := count_tbl.first;
  loop
    key_value.key := i;
    key_value.value := count_tbl(i);
    pipe row(key_value);
    i := count_tbl.next(i);
    exit when i is null;
  end loop;
  return;
end reduce;

end map_and_reduce;
/
```

パラレルパイプライン表関数実行例

```
SQL> select * from input_tbl;
DATA
--------------------------------------------------------------------------------

Hello world!
Hello Oracle!
SQL> select * from table(map_and_reduce.reduce(cursor(select * from
table(map_and_reduce.map(cursor(select * from input_tbl))))));
KE      VALUE
-- ----------
            2
!           2
H           2
O           1
a           1
c           1
d           1
e           3
l           6
o           3
r           2
w           1

12行が選択されました。
```

パラレルパイプライン表関数 コーディングポイント(その1)

- 基本はパイプライン表関数と同じ
  - PIPELINE句の付与
  - PIPE ROW文によるレコード単位の出力
  - 変数指定なしのRETURN文
- 引数に最低一つのカーソル変数(REF CURSOR)が必要
  - 複数以上のカーソル変数の指定も可能
  - カーソル変数以外のデータ型の引数の指定も可能

```
function map (p_cur in ref cur clob)
  return key_value_type_tbl pipelined
…
```

事前定義したカーソル変数

SYS_REFCURSOR型も指定可

パラレルパイプライン表関数 コーディングポイント(その2)

- データのストリーム化方法
  - フェッチしたデータのハンドリング方法
  - 以下の3種類のいずれか指定可能
    1. 指定なし ： フェッチ後何もしない
    2. CLUSTER ： フェッチ後指定キーの値にてグルーピングする
    3. ORDER ： フェッチ後指定キー(複数指定可)で並べ替える
  - 処理ロジック上、処理スピードやコーディングしやすさの観点で有利に場合に指定

```
function reduce (p_cur in ref_cur_kv)
  return key_value_type_tbl pipelined
  cluster p_cur by (key)
...
```

カーソル変数の引数名

グルーピング対象列

```
function reduce (p_cur in ref_cur_kv)
  return key_value_type_tbl pipelined
  order p_cur by (key, value)
...
```

カーソル変数の引数名

並べ替え対象列

パラレルパイプライン表関数 コーディングポイント(その3-1)

- データのパーティション化方法
  - 指定必須(PARALLEL_ENABLE句)
  - 並列処理する際の、分割基準を指定
  - 並列度の指定は不可
  - 分割基準として、以下の3種類のいずれか指定可能
    1. ANY　:　Oracleまかせ
    2. HASH　:　指定キーのハッシュ値で分割
    3. RANGE　:　指定キー(複数指定可)の範囲で分割

パラレルパイプライン表関数 コーディングポイント(その3-2)

- データのパーティション化方法 ： ANY
  - 分割をOracleまかせにする
  - 入力が1行で分割のしようがない場合に指定
    - 1行から複数行を生成するアプリロジックに向いている
    - 上記以外のほとんどの場合もANYでOK

```
function reduce (p_cur in ref_cur_kv)
  return key_value_type_tbl pipelined
  cluster p_cur by (key)
parallel_enable(partition p_cur by any)
is
...
```

カーソル変数の引数名

パラレルパイプライン表関数 コーディングポイント(その3-3)

- データのパーティション化方法 ： HASH
  - 指定キーのハッシュ値で分割

```
function reduce (p_cur in ref_cur_kv)
  return key_value_type_tbl pipelined
  order p_cur by (key)
parallel_enable(partition p_cur by hash(key))
is
...
```

カーソル変数の引数名

ハッシュ化対象列

- データのパーティション化方法 ： RANGE
  - 指定キー(複数指定可)の範囲で分割

```
function reduce (p_cur in ref_cur_kv)
  return key_value_type_tbl pipelined
  cluster p_cur by (key)
parallel_enable(partition p_cur by range(key, value))
is
...
```

カーソル変数の引数名

範囲化対象列

Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix
  - Windows RAC環境にてUNCアクセスを可能にする手順
  - ネイティブコンパイル設定手順(Oracle9i)
  - ネイティブコンパイル設定手順(Oracle10g)
  - OracleとHadoopの連携

Windows RAC環境にてUNCアクセスを可能にする手順

1. 全ノードにサービス起動用OSアカウントを作成
   - Administrators および ORA_DBA グループに所属させる
   - 全ノードで同じパスワードを設定
2. 全ての CRS リソースを srvctl コマンドを用いて停止

```
srvctl stop database -d <データベース名>
srvctl stop instance -i <インスタンス名> -d <データベース名>
srvctl stop nodeapps -n <ノード名>
```

3. 全ノードにおいてcrsuser コマンドを実行してOracleCRSToken_<OSアカウント名> サービスを作成
   - %ORA_CRS_HOME%¥bin¥crsuser add <OSアカウント名>
   - パスワードは何も入力せずに ENTER を実行
   - <OSアカウント名> には、以下のとおり指定
     - ローカルユーザーの場合: コンピュータ名¥ユーザー名
     - ドメインユーザーの場合: ドメイン名¥ユーザー名
   - エラーで終了しても、OracleCRSToken_<OSアカウント名> サービスが作成されていればOK

Windows RAC環境にてUNCアクセスを可能にする手順

4. 3 で追加した OS アカウントが正しく登録されているかをcrsuser listコマンドで確認

```
C:¥> crsuser list
Available nodes
            myrac1
            myrac2
Properly configured users
            testuser
```

5. 全ノードにおいて OracleCRSToken_<OSアカウント名> サービスのログオンアカウントに手動でパスワードを設定してサービスを開始
6. 全ノードにおいて OracleService<SID>サービス, OracleTNSListener<Listener名>サービスが停止されていることを確認し、サービスのログオンアカウントを変更
   - サービスが起動している場合は変更前に停止すること
   - ASMインスタンスのサービスのログオンアカウントは変更しないこと

Windows RAC環境にてUNCアクセスを可能にする手順

7. 任意の1ノードにおいて、%ORA_CRS_HOME%¥bin¥crs_setperm コマンドを使用し、リスナー、インスタンス、データベースリソースの OWNER を変更
   - crs_setperm <リソース名> -o <OSアカウント名>
   - ASM が存在する場合は ASM リソースについても変更
   - <OSアカウント名> には OracleService<SID> に指定予定のアカウントを指定
     - コンピュータ名、ドメイン名の指定は不要

```
C:¥> crs_setperm ora.rac102.db -o testuser
C:¥> crs_setperm ora.rac102.rac1021.inst -o testuser
C:¥> crs_setperm ora.rac102.rac1022.inst -o testuser
C:¥> crs_setperm ora.myrac1.LISTENER_MYRAC1.lsnr -o testuser
C:¥> crs_setperm ora.myrac2.LISTENER_MYRAC2.lsnr -o testuser
```

8. %ORA_CRS_HOME%¥bin¥crs_getperm コマンドを使用し、変更が反映されていることを確認

```
C:¥> crs_getperm ora.rac102.db
Name: ora.rac102.db
owner:testuser:rwx,pgrp::---,other::r--,
```

9. srvctl コマンドを用いて各リソースを起動

```
srvctl start nodeapps -n <ノード名>
srvctl start instance -i <インスタンス名> -d <データベース名>
srvctl start database -d <データベース名>
```

Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix
  - Windows RAC環境にてUNCアクセスを可能にする手順
  - ネイティブコンパイル設定手順(Oracle9i)
  - ネイティブコンパイル設定手順(Oracle10g)
  - OracleとHadoopの連携

ネイティブコンパイル設定手順(Oracle9i)

1. makeファイルを必要に応じて編集
   - $ORACLE_HOME/plsql/spnc_makefile.mk
   - Solarisは最低限ccへのパスを修正する必要あり
2. 共有ライブラリ格納用ディレクトリを作成
3. 以下の初期化パラメータを設定
   - ① PLSQL_NATIVE_MAKE_FILE_NAME
     - 1.のフルパス名称
   - ② PLSQL_NATIVE_MAKE_UTILITY
     - makeコマンドのフルパス名称
   - ③ PLSQL_NATIVE_LIBRARY_DIR
     - 2.のフルパス名称
   - ④ (PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT)
     - コンパイル対象モジュールが多い場合に指定すると、ファイルが特定ディレクトリに集中しないようディレクトリ分割を自動で行う

ネイティブコンパイル設定手順(Oracle9i)

1. コンパイルを実施するユーザーでログインし、初期化パラメータPLSQL_COMPILER_FLAGSを「NATIVE」に設定
   - デフォルトは「INTERPRETED」(PL/SQLバイトコード形式にコンパイル)
   - SQL> alter session set plsql_compiler_flags = 'NATIVE';共有ライブラリ格納用ディレクトリを作成
2. PL/SQLソースをコンパイル
   - SQL> create or replace procedure …;

Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix
  - Windows RAC環境にてUNCアクセスを可能にする手順
  - ネイティブコンパイル設定手順(Oracle9i)
  - ネイティブコンパイル設定手順(Oracle10g)
  - OracleとHadoopの連携

ネイティブコンパイル設定手順(Oracle10g)

1. コンパイル実行ファイルを必要に応じて編集
   - $ORACLE_HOME/plsql/spnc_commands
2. 共有ライブラリ格納用ディレクトリを作成
3. 以下の初期化パラメータを設定(設定内容は9iの時と同じ)
   - PLSQL_NATIVE_LIBRARY_DIR
   - (PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT)
4. コンパイルを実施するユーザーでログインし、初期化パラメータPLSQL_CODE_TYPEを「NATIVE」に設定
   - デフォルトは「INTERPRETED」(PL/SQLバイトコード形式にコンパイル)
   - SQL> alter session set plsql_code_type = 'NATIVE';
5. PL/SQLソースをコンパイル
   - SQL> create or replace procedure …;

Agenda

- あまり語られないUTL_FILEの仕様
- ソースを直さず高速化
- PL/SQLのパラレル実行～PL/SQLでMapReduce?～
- Appendix
  - Windows RAC環境にてUNCアクセスを可能にする手順
  - ネイティブコンパイル設定手順(Oracle9i)
  - ネイティブコンパイル設定手順(Oracle10g)
  - OracleとHadoopの連携

Oracle DatabaseとHadoopの連携(その1)

- Oracle Loader for Hadoop
  - Hadoop上のデータをOracle Databaseに効率的にロード
  - ロードの際にMapReduce処理が可能
- Oracle Data Integrator Application Adapter for Hadoop
  - Oracle Data Integrator(ETLツール)でHadoopのデータを扱えるようにするためのアダプタ

Oracle DatabaseとHadoopの連携(その2)

- Oracle Big Data Appliance
  - 以下のソフトウェアをプリインストール
    - Oracle Hadoop
    - Oracle Loader for Hadoop
    - Oracle Data Integrator
      (with Application Adapter for Hadoop)
    - Oracle NoSQL Database
  - 大容量メモリ、Infiniband結線などによる高速化
  - Oracle ExadataともInfiniband経由で連携

OTNセミナーオンデマンド

コンテンツに対する
ご意見・ご感想を是非お寄せください。

OTNオンデマンド　感想

http://blogs.oracle.com/oracle4engineer/entry/otn_ondemand_questionnaire

上記に簡単なアンケート入力フォームをご用意しております。

**セミナー講師/資料作成者にフィードバックし、
コンテンツのより一層の改善に役立てさせていただきます。**

是非ご協力をよろしくお願いいたします。

OTNセミナーオンデマンド

日本オラクルのエンジニアが作成したセミナー資料・動画ダウンロードサイト

ORACLE TECHNOLOGY NETWORK
OTNセミナー オンデマンド

掲載コンテンツカテゴリ(一部抜粋)
Database 基礎
Database 現場テクニック
Database スペシャリストが語る
Java
WebLogic Server/アプリケーション・グリッド
EPM/BI 技術情報
サーバー
ストレージ

100以上のコンテンツをログイン不要でダウンロードし放題

データベースからハードウェアまで充実のラインナップ

毎月、旬なトピックの新作コンテンツが続々登場

例えばこんな使い方

- 製品概要を効率的につかむ
- 基礎を体系的に学ぶ/学ばせ
- 時間や場所を選ばず（オンデマンド講
- スマートフォンで通勤中にも受講可能

毎月チェック！

コンテンツ一覧 はこちら
http://www.oracle.com/technetwork/jp/ondemand/index.html

新作＆おすすめコンテンツ情報 はこちら
http://oracletech.jp/seminar/recommended/000073.html

OTNオンデマンド

ORACLE

オラクルエンジニア通信

オラクル製品に関わるエンジニアの方のための技術情報サイト

技術資料 — インストールガイド・設定チュートリアルetc. 欲しい資料への最短ルート

アクセスランキング — 他のエンジニアは何を見ているのか？人気資料のランキングは毎月更新

特集テーマ Pick UP — 性能管理やチューニングなど月間テーマを掘り下げて詳細にご説明

技術コラム — SQLスクリプト、索引メンテナンスetc. 当たり前の運用/機能が見違える!?

http://blogs.oracle.com/oracle4engineer/

オラクルエンジニア通信

oracletech.jp

ITエンジニアの皆様に向けて旬な情報を楽しくお届け

製品/技術情報
Oracle Databaseっていくら？オプション機能も見積れる簡単ツールが大活躍

セミナー
基礎から最新技術までお勧めセミナーで自分にあった学習方法が見つかる

スキルアップ
ORACLE MASTER！試験頻出分野の模擬問題と解説を好評連載中

Viva! Developer
全国で活躍しているエンジニアにスポットライト。きらりと輝くスキルと視点を盗もう

http://oracletech.jp/

oracletech

あなたにいちばん近いオラクル

Oracle Direct

まずはお問合せください

Oracle Direct

システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。
ステム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

※フォームの入力にはログインが必要となります。
※こちらから詳細確認のお電話を差し上げる場合がありますので
ご登録の連絡先が最新のものになっているかご確認下さい。

フリーダイヤル

0120－155－096

※月曜～金曜
9:00～12:00、13:00～18:00
（祝日および年末年始除く）

ORACLE

Hardware and Software
Engineered to Work Together


ORACLE®