Oracle8*i*

NLS ガイド

リリース 8.1

ORACLE

Oracle8*i* NLS ガイド リリース 8.1

部品番号 : A62779-1

第 1 版 : 1999 年 5 月（第 1 刷）

原本名 : Oracle8*i* National Language Support Guide, Release 8.1.5

原本部品番号 : A67789-01

原本著者 : Paul Lane、Gail Yamanaka

原本協力者 : Winson Chu、Sandy Dreskin、Jason Durbin、Jessica Fan、Yu Gong、Josef Hasenberger、Claire Ho、Lefty Leverenz、Tom Portfolio、Den Raphaely、Makoto Tozawa、Hiro Yoshioka

グラフィック・デザイナ : Valarie Moore

目次

2 NLS 環境の設定

3　キャラクタ・セットの選択

4 SQL プログラミング

5 OCI プログラミング

A ロケール・データ

B ロケール・データのカスタマイズ

C 廃止されたロケール・データ

D 用語集

索引

はじめに

このマニュアルでは、Oracle の各国語サポート（NLS）機能に関する情報について説明します。次のような参照情報を記載しています。

- Oracle NLS の理解
- NLS 環境の設定
- キャラクタ・セットの選択
- SQL プログラミング
- OCI プログラミング
- ロケール・データ
- ロケール・データのカスタマイズ
- 廃止されたロケール・データ
- 用語集

機能の適用範囲と可用性

『Oracle8*i* NLS ガイド』では、複数の言語またはキャラクタ・セットを使用する環境で作業する場合に共通する多くの問題の対処方法について説明します。

『Oracle8*i* NLS ガイド』では、Oracle8*i* と Oracle8*i* Enterprise Edition の両方の製品に共通する機能について説明します。

対象読者

このマニュアルは、NLS に関する問題を処理する必要があるデータベース管理者、システム管理者およびデータベース・アプリケーション開発者を対象にしています。

前提条件

このマニュアルの読者は、リレーショナル・データベースの概要、Oracle Server の基本概念、および Oracle を稼働しているオペレーティング・システム環境について理解している必要があります。

インストレーションおよび移行に関する情報

このマニュアルではインストレーションや移行については説明しません。したがって、インストレーションについての情報が必要な場合は、オペレーティング・システム固有の Oracle ドキュメントを参照してください。データベースおよびアプリケーションの移行については、『Oracle8*i* 移行ガイド』を参照してください。

アプリケーションの設計に関する情報

管理者だけでなく、Oracle の使用経験が豊富なユーザーや上級データベース・アプリケーション・デザイナにとっても、このマニュアルの情報は役に立ちます。ただし、データベース・アプリケーション開発者は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』、および Oracle データベース・アプリケーションの開発に使用するツールまたは言語製品のドキュメントも参照してください。

Oracle8*i* NLS ガイドの構成

このマニュアルは次のように構成されています。

第 1 章「Oracle NLS の理解」

NLS 問題の概要と Oracle での NLS の使用方法について説明します。

第 2 章「NLS 環境の設定」

Oracle の NLS 機能について説明します。

第 3 章「キャラクタ・セットの選択」

NLS 機能を利用する場合の使用例について説明します。

第 4 章「SQL プログラミング」

SQL プログラミングでの NLS 考慮事項について説明します。

第 5 章「OCI プログラミング」

OCI プログラミングでの NLS 考慮事項について説明します。

付録 A「ロケール・データ」

Oracle Server でサポートされている言語、地域、キャラクタ・セットおよびその他のロケール・データについて説明します。

付録 B「ロケール・データのカスタマイズ」

NLS データ・オブジェクトのカスタマイズ方法について説明します。

付録 C「廃止されたロケール・データ」

廃止されたキャラクタ・セット名の一部を示します。

付録 D「用語集」

NLS 用語の定義について説明します。

表記上の規則

次の項では、このマニュアルで使用している規則について説明します。

本文

このマニュアルの本文では、次の規則が使用されています。

大文字

大文字は、コマンドのキーワード、データベース・オブジェクト名、パラメータ、ファイル名などを明示するために使用されています。

たとえば、「デフォルト値を挿入した後に、DEPTNO 列に定義されている FOREIGN KEY 整合性制約がチェックされます」、「プライベート・ロールバック・セグメントを作成する場合、ロールバック・セグメントの名前は、ROLLBACK_SEGMENTS 初期化パラメータに指定しなければなりません。」などです。

イタリック表記の文字

文中のイタリック表記は、ユーザーが入力する値を示します。

コード例

SQL のコマンドまたは文は、Oracle Enterprise Manager ライン・モード（Server Manager）および SQL*Plus では、クーリエ・フォントで表示されます。

次に例を示します。

```
INSERT INTO emp (empno, ename) VALUES (1000, 'SMITH');
ALTER TABLESPACE users ADD DATAFILE 'users2.ora' SIZE 50K;
```

例文には、カンマ、引用符などの句読点が含まれている場合があります。例文に示されている句読点はすべて必須です。すべての例文はセミコロン（;）で終了しています。アプリケーションによっては、文を終了するためにセミコロンまたはその他の終了文字が必要となります（必要でない場合もあります）。

コード例での大文字

例文では、大文字によって Oracle SQL 内のキーワードを示しています。ただし、文を発行する場合、キーワードの大 / 小文字は区別されません。

コード例での小文字

例文では、小文字によってその単語が単なる例として使用されていることを示します。たとえば、小文字の単語は、表、列またはファイルの名前を示します。

1

Oracle NLS の理解

この章では、次の Oracle NLS の概要について説明します。

- Oracle Server の NLS アーキテクチャ
- 標準機能
- カスタマイズ機能
- SQL サポート

Oracle Server の NLS アーキテクチャ

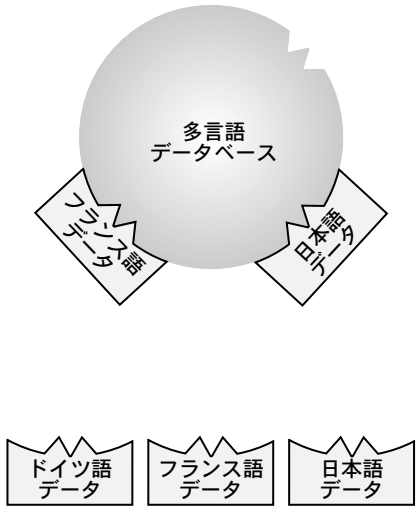
Oracle の各国語サポート（NLS）アーキテクチャによって、母国語でデータを格納、処理、および検索することができます。データベースのユーティリティやエラー・メッセージ、ソート順、日付、時刻、通貨単位、数値およびカレンダの規則には、母国語やロケールの設定が自動的に適応されます。

個々の規則がどのように動作するかは、パラメータの設定によって決まります。

ロケールに依存しない操作

Oracle の各国語サポート・アーキテクチャは、Oracle の NLS ランタイム・ライブラリとともにインプリメントされます。NLS ランタイム・ライブラリは、広範囲にわたる言語非依存の関数パッケージを提供します。これによって、テキストと文字を適切に処理し、言語上の規則に従って操作できるようになります。これらの機能は、特定の言語およびロケールのためのもので、実行時に特定されロードされたロケール固有データ・セットによって制御されます。

次の図に、実行時のロケール固有データのロードについて示します。例として、フランス語および日本語のロケール・データがロードされています。



ロケール固有の NLS データは、環境変数 ORA_NLS* で指定したディレクトリに格納されます。それぞれの新規リリースによって、対応する ORA_NLS データ・ディレクトリは異なります。Oracle8*i* リリース 8.1 の場合、ORA_NLS33 ディレクトリが使用されます。たとえば、ほとんどの UNIX プラットフォームでは、環境変数 ORA_NLS33 に **$ORACLE_HOME/ocommon/nls/admin/data** を設定する必要があります。

表 1-1 NLS データの位置

リリース	環境変数
7.2	ORA_NLS
7.3	ORA_NLS32
8.0、8.1	ORA_NLS33

複雑な Oracle 環境でシステムを実行している場合は、変数 ORA_NLS* が適切に設定され、そのリリースに対応する NLS データ・ファイルが使用可能であることを確認してください。

ブート・ファイルは、ロードできる NLS オブジェクトの可用性を判断するために使用されます。Oracle では、システムおよびユーザーの両方のブート・ファイルをサポートします。ユーザー・ブート・ファイルを使用すると、そのデータベースで使用可能な NLS ロケール・オブジェクトを柔軟に調整でき、これによって、メモリーの消費を制御できます。さらに、

新しいロケール・データを追加したり、ロケール・データ・コンポーネントをカスタマイズしたりすることもできます。

クライアント / サーバー・アーキテクチャ

Oracle8*i* は、クライアント / サーバー・アーキテクチャを使用してインプリメントされます。言語に依存する操作は、クライアントおよびサーバーの両方にあるたくさんのパラメータおよび環境変数によって制御されます。サーバーおよびクライアントは、同じまたは異なるロケールで実行されている場合があり、同じまたは異なる言語要件が指定されています。クライアントとサーバーで異なるキャラクタ・セットが指定されている場合、Oracle8*i* では、文字列に関するキャラクタ・セット変数が自動的に使用されます。

標準機能

Oracle の標準機能では、さまざまな日付、時刻、カレンダ、通貨単位、数値およびキャラクタ・セットの書式と同様に、言語および地域もサポートします。

言語サポート

Oracle8*i* を使用すると、ユーザーは母国語でデータを格納、処理および検索することができます。表 1-2「言語サポート」に、サポートしている言語を示します。アスタリスクは、製品自体がその言語で翻訳されていることを示します。

表 1-2 言語サポート

American English *	English	Japanese *	Simplified Chinese *
Arabic *	Estonian	Korean *	Slovak *
Bengali	Finnish *	Latin American Spanish *	Slovenian
Brazilian Portuguese *	French *	Latvian	Spanish *
Bulgarian	German	Lithuanian	Swedish *
Canadian French	German Din	Malay	Thai
Catalan *	Greek *	Mexican Spanish	Traditional Chinese *
Croatian	Hebrew *	Norwegian *	Turkish *
Czech *	Hungarian *	Polish *	Ukrainian
Danish *	Icelandic	Portuguese *	Vietnamese
Dutch *	Indonesian	Romanian *	
Egyptian	Italian *	Russian *	

Oracle がサポートする言語の名前および略称のすべてのリストについては、A-2 ページの「言語」を参照してください。

メッセージ・サポート

ユーティリティおよびエラー・メッセージは、母国語で表示できます。

地域サポート

Oracle8*i* では、指定された地理上の場所によって固有の異なる文化的規則をサポートしています。ローカル時間、日付、数値および通貨単位に関する規則が処理されます。次に、サポートしている地域を示します。

表 1-3 地域サポート

Algeria	Egypt	Latvia	Spain
America	Estonia	Lithuania	Sudan
Austria	Finland	Luxembourg	Sweden
Australia	France	Malaysia	Switzerland
Bahrain	Germany	Mauritania	Syria
Bangladesh	Greece	Mexico	Taiwan
Belgium	Hong Kong	Morocco	Thailand
Brazil	Hungary	New Zealand	The Netherlands
Bulgaria	Iceland	Norway	Tunisia
Canada	Indonesia	Oman	Turkey
Catalonia	Iraq	Poland	Ukraine
China	Ireland	Portugal	United Arab Emirates
CIS	Israel	Qatar	United Kingdom
Croatia	Italy	Romania	Uzbekistan
Cyprus	Japan	Saudi Arabia	Vietnam
Czech Republic	Jordan	Slovakia	Yemen
Czechoslovakia	Kazakhstan	Slovenia	
Denmark	Korea	Somalia	
Djibouti	Kuwait	South Africa	

日付と時刻の書式

世界中で使用されているさまざまな時、日、月および年に関する規則が、ローカル書式で処理されます。

通貨単位と数値の書式

通貨、貸方および借方の記号は、ローカル書式で表現されます。基数記号と 3 桁区切りは、ロケールで定義します。

カレンダ

Gregorian（グレゴリオ暦）、Japanese Imperial（日本の元号制）、ROC Official（台湾暦）、Thai Buddha（タイ仏教暦）、Persian（ペルシャ暦）、English Hijrah（英語版イスラム紀元）、および Arabic Hijrah（イスラム紀元）をサポートしています。カレンダの全リストについては、A-19 ページの「暦法」を参照してください。

言語ソート

Oracle8*i* では、文化的に正確なソートを行うために言語ソートが提供されています。

表 1-4 言語の定義

基本名	拡張名	特殊な事例
ARABIC	--	
ARABIC_MATCH	--	
ARABIC_ABJ_SORT	--	
ARABIC_ABJ_MATCH	--	
ASCII7	--	
BENGALI	--	
BULGARIAN	--	
CANADIAN FRENCH	--	
CATALAN	XCATALAN	æ、AE、ß
CROATIAN	XCROATIAN	D、L、N、d、l、n、ß

表 1-4 言語の定義

基本名	拡張名	特殊な事例
CZECH	XCZECH	ch、CH、Ch、ß
DANISH	XDANISH	A、ß、Å、å
DUTCH	XDUTCH	ij、IJ
EEC_EURO	--	
EEC_EUROPA3	--	
ESTONIAN	--	
FINNISH	--	
FRENCH	XFRENCH	
GERMAN	XGERMAN	ß
GERMAN_DIN	XGERMAN_DIN	ß、ä、ö、ü、Ä、Ö、Ü
GREEK	--	
HEBREW	--	
HUNGARIAN	XHUNGARIAN	cs、gy、ny、sz、ty、zs、ß、CS、Cs、GY、Gy、NY、Ny、SZ、Sz、TY、Ty、ZS、Zs
ICELANDIC	--	
INDONESIAN	--	
ITALIAN	--	
JAPANESE	--	
LATIN	--	
LATVIAN	--	
LITHUANIAN	--	
MALAY	--	
NORWEGIAN	--	
POLISH	--	
PUNCTUATION	XPUNCTUATION	
ROMANIAN	--	
RUSSIAN	--	

表 1-4 言語の定義

基本名	拡張名	特殊な事例
SLOVAK	XSLOVAK	dz、DZ、Dz、ß（*caron*）
SLOVENIAN	XSLOVENIAN	ß
SPANISH	XSPANISH	ch、ll、CH、Ch、LL、Ll
SWEDISH	--	
SWISS	XSWISS	ß
THAI_DICTIONARY	--	
THAI_TELEPHONE	--	
TURKISH	XTURKISH	æ、AE、ß
UKRAINIAN	--	
UNICODE_BINARY		
VIETNAMESE	--	
WEST_EUROPEAN	XWEST_EUROPEAN	ß

キャラクタ・セット・サポート

Oracle では、各国、国際およびベンダー固有の規格に基づいた、たくさんのシングルバイト、マルチバイトおよび固定幅コード体系をサポートしています。サポートしているキャラクタ・セットの全リストについては、A-6 ページの「キャラクタ・セット」を参照してください。

カスタマイズ機能

Oracle を使用すると、キャラクタ・セットおよびカレンダをカスタマイズできます。

キャラクタ・セットのカスタマイズ

特殊記号、ベンダー固有の文字、または固有名、履歴用語などを表す文字をサポートするために、ユーザー定義文字が必要な場合があります。開発者は、Unicode Private Use Area を使用して既存のキャラクタ・セットの定義を拡張できます。詳細は、B-1 ページの「カスタマイズ済みキャラクタ・セット」を参照してください。

カレンダのカスタマイズ

定規年号および太陰暦の偏差日を定義できます。詳細は、B-11 ページの「カスタマイズ済みカレンダ」を参照してください。

SQL サポート

NLS パラメータは、SQL 関数の動作を変更するために使用できます。たとえば、SQL 関数がソート、文字分類、時刻、日付、通貨単位および数値の書式を処理する場合、ユーザー環境に暗黙に設定された、または関数コールでのパラメータとして明示的に設定されたさまざまな NLS パラメータに基づいて、その動作を変更できます。関数コールの詳細は第 4 章「SQL プログラミング」を、環境パラメータの詳細は第 2 章「NLS 環境の設定」を参照してください。

2

NLS 環境の設定

この章では、NLS 環境の設定方法について説明します。トピックは次のとおりです。

NLS パラメータの設定

NLS パラメータによって、クライアントとサーバーの両方でのロケール固有の動作が決定します。NLS パラメータは、次の 4 通りの方法で指定できます。

1. サーバーの初期化パラメータとして指定します。パラメータは、デフォルト・サーバーの NLS 環境を指定する初期化ファイル（INIT.ORA）に挿入します。これらの設定は、クライアント側には影響しません。サーバーの動作のみを制御します。たとえば、次のように指定します。

```
NLS_TERRITORY = "CZECH REPUBLIC"
```

2. クライアントの環境変数として指定します。NLS パラメータは、ロケールに依存する動作をクライアントに指定するために使用されます。その結果、初期化ファイル内でサーバーに設定されたデフォルトは無効になります。たとえば、UNIX システム上で次のように指定します。

```
% setenv NLS_SORT FRENCH
```

3. ALTER SESSION パラメータとして指定します。ALTER SESSION 文で設定した NLS パラメータによって、初期化ファイル内でサーバーに設定されたデフォルト、またはクライアントで環境変数に設定されたデフォルトは無効になります。

```
SVRMGR> ALTER SESSION SET NLS_SORT = FRENCH
```

ALTER SESSION の詳細は、『Oracle8*i* SQL リファレンス』を参照してください。

4. SQL 関数のパラメータとして指定します。NLS パラメータを明示的に使用して、SQL 関数内での NLS 動作をハードコードできます。これによって、初期化ファイル内でサーバーに設定されたデフォルト、クライアントで環境変数に設定されたデフォルト、またはクライアントでの ALTER SESSION で設定されたデフォルトは無効になります。たとえば、次のように指定します。

```
TO_CHAR(hiredate, 'DD/MON/YYYY', 'nls_date_language = FRENCH')
```

表 2-1 に、NLS パラメータを使用する場合の優先順位を示します。優先順位が低い設定は、優先順位がより高い設定で上書きされます。たとえば、優先順位が最も低いと予測できるデフォルト値は、他のすべての方法で上書きできます。また、SQL 関数内で NLS パラメータを明示的に設定すると、他のすべての設定（デフォルト、INIT.ORA、環境変数および ALTER SESSION パラメータ）を上書きできます。

表 2-1 パラメータと優先順位

	最も高い優先順位
1	SQL 関数での明示的な設定
2	ALTER SESSION 文での設定
3	環境変数としての設定
4	初期化パラメータ・ファイル内での指定
5	デフォルト
	最も低い優先順位

表 2-2 に、Oracle Server で使用できる NLS パラメータを示します。

表 2-2 パラメータと有効範囲

パラメータ	説明	デフォルト	有効範囲 (I=INIT.ORA、 E= 環境変数、 A=ALTER SESSION)
NLS_CALENDAR	暦法	Gregorian (グレゴリオ暦)	I、-、A
NLS_COMP	SQL 演算子の比較	Binary	-、E、A
NLS_CREDIT	貸方の会計記号	NLS_TERRITORY	I、E、A
NLS_CURRENCY	各国通貨記号	NLS_TERRITORY	I、E、A
NLS_DATE_FORMAT	日付書式	NLS_TERRITORY	I、E、A
NLS_DATE_LANGUAGE	曜日および月の名前に使用する言語	NLS_LANGUAGE	I、E、A
NLS_DEBIT	借方の会計記号	NLS_TERRITORY	I、E、A
NLS_ISO_CURRENCY	ISO 国際通貨記号	NLS_TERRITORY	I、E、A
NLS_LANG	言語、地域、キャラクタ・セット	American_ America.US7ASCII	-、E、-
NLS_LANGUAGE	言語	NLS_LANG	I、-、A
NLS_LIST_SEPARATOR	リスト内での項目の区切り文字	NLS_TERRITORY	I、-、A
NLS_MONETARY_CHARACTERS	ドルおよびセントに使用する通貨記号 (または相当する記号)	NLS_TERRITORY	I、E、A
NLS_NCHAR	各国キャラクタ・セット	NLS_LANG	-、E、-
NLS_NUMERIC_CHARACTERS	小数点文字およびグループ・セパレータ	NLS_TERRITORY	I、E、A
NLS_SORT	文字のソート基準	NLS_LANGUAGE	I、E、A
NLS_TERRITORY	地域	NLS_LANG	I、-、A
NLS_DUAL_CURRENCY	第二通貨記号	NLS_TERRITORY	I、E、A

NLS_LANG を使用したロケールの選択

ロケールとは、システムまたはプログラムを実行する言語的および文化的環境のことです。ロケール動作を指定するには、NLS_LANG パラメータの設定が最も簡単な方法です。NLS_LANG を設定すると、データベースで（サーバー・セッションとクライアント・アプリケーションの両方について）使用する言語、地域およびキャラクタ・セットが設定されます。1

つのパラメータを使用すると、確実に、サーバーとクライアントの両方の言語および地域環境が同じになります。

NLS_LANG パラメータの書式には、3 つのコンポーネント（*language*、*territory* および *charset*）があります。

```
NLS_LANG = language_territory.charset
```

それぞれのコンポーネントは、NLS 機能のサブセットの動作を制御します。

language	Oracle メッセージ、曜日および月の名前に使用する言語などの規則を指定します。サポートしているそれぞれの言語には、American（米語）、French（フランス語）、German（ドイツ語）などの固有の名前があります。言語引数によって、地域およびキャラクタ・セットの引数のデフォルト値が指定され、その結果、territory または charset のいずれか（あるいはその両方）を省略できます。language を指定しない場合、デフォルトでは American になります。言語の全リストについては、「言語」を参照してください。
territory	デフォルトのカレンダ、照合、日付、通貨単位および数値書式などの規則を指定します。サポートしているそれぞれの地域には、America（アメリカ）、France（フランス）、Canada（カナダ）などの固有の名前があります。territory を指定しない場合、デフォルトでは America になります。地域の全リストについては、「地域」を参照してください。
charset	クライアント・アプリケーションが使用するキャラクタ・セット（通常はユーザー端末で使用するキャラクタ・セット）を指定します。サポートしているそれぞれのキャラクタ・セットには、US7ASCII、WE8ISO8859P1、WE8DEC、WE8EBCDIC500、JA16EUC などの一意の頭字語があります。それぞれの言語には、デフォルトのキャラクタ・セットが対応付けられています。システムで使用可能な言語のデフォルト値については、インストレーション・ガイドまたはユーザーズ・ガイドに説明があります。キャラクタ・セットの全リストについては、「キャラクタ・セット」を参照してください。

注意：NLS_LANG 定義のコンポーネントはすべてオプションです。特に指定しない項目はデフォルトになります。*territory* または *charset* を指定する場合、先行デリミタを付ける必要があります。先行デリミタは、*territory* の場合はアンダースコア（_）で、*charset* の場合はピリオド（.）です。先行デリミタを付けないと、値は言語名として解析されます。

NLS_LANG の 3 つの引数は、次の例のように、さまざまな組合せで指定できます。

```
NLS_LANG = AMERICAN_AMERICA.US7ASCII
```

または

```
NLS_LANG = FRENCH_CANADA.WE8DEC
```

または

```
NLS_LANG = JAPANESE_JAPAN.JA16EUC
```

非論理的な組合せでも設定できますが、正しく動作しません。たとえば、次のように西ヨーロッパ諸国のキャラクタ・セットを使用して日本語をサポートするとします。

```
NLS_LANG = JAPANESE_JAPAN.WE8DEC
```

WE8DEC は日本語の文字をサポートしないので、その結果、日本語データを格納できなくなります。

NLS_LANG の指定

NLS_LANG を環境変数として、コマンド行で設定できます。たとえば、UNIX 上では、次の行をプロンプトで入力して、NLS_LANG の値を指定できます。

```
% setenv NLS_LANG FRENCH_FRANCE.WE8DEC
```

NLS_LANG の例

NLS_LANG は環境変数であるため、クライアント・アプリケーションによって起動時に読み込まれます。クライアントは、NLS_LANG で定義された情報を、接続時にサーバーへ送ります。

次に、日付および数値の書式が NLS_LANG によってどのように影響を受けるかを示します。

```
%seenv NLS_LANG American_America.WE8ISO8859P1
SVRMGR> SELECT ename, hiredate, ROUND(sal/12,2)  sal FROM emp;
ENAME                          HIREDATE            SAL
---------                      ---------           ------
Clark                          09-DEC-88           4195.83
Miller                         23-MAR-92           4366.67
Strauß                         01-APR-95           3795.87
```

NLS_LANG で、言語にフランス語、地域にフランス、キャラクタ・セットに西欧の 8 ビット ISO 8859-1 を設定した場合も、同じ問合せを実行します。

```
%setenv NLS_LANG French_France.WE8ISO8859P1;
SVRMGR> SELECT ename, hiredate, ROUND(sal/12,2)  sal FROM emp;
ENAME                          HIREDATE            SAL
---------                      ---------           -------
Clark                          09/12/88            4195,83
Miller                         23/03/92            4366,67
Strauß                         01/04/95            3795,87
```

言語および地域指定の変更

NLS_LANG は、データベースで（サーバー・セッションとクライアント・アプリケーションの両方について）使用する NLS 言語および地域環境を設定します。1 つのパラメータを使用すると、確実に、データベースおよびクライアント・アプリケーションの両方の言語環境が自動的に同じになります。ご使用の環境を変更することが必要な場合もあります。変更する場合は、NLS_LANGUAGE または NLS_TERRITORY を使用してください。

NLS_LANGUAGE

パラメータ・タイプ :	文字列
パラメータの有効範囲 :	初期化パラメータおよび ALTER SESSION
デフォルト値 :	NLS_LANG
値の範囲 :	有効な言語名

NLS_LANGUAGE は、次のセッション特性に対するデフォルトの規則を指定します。

- サーバー・メッセージの言語
- 曜日と月の言語、およびその略語（SQL 関数 TO_CHAR と TO_DATE で指定されます）
- AM、PM、AD および BC に相当する記号
- ORDER BY を指定する際の文字データに対するデフォルトのソート基準（ORDER BY を指定しない限り、GROUP BY はバイナリ・ソートを使用します）
- 書込み方向
- 肯定的 / 否定的応答文字列

初期化ファイルの中で NLS_LANGUAGE に指定した値は、そのインスタンスのすべてのセッションに対してデフォルト値になります。

たとえば、デフォルトのセッション言語をフランス語に指定するには、パラメータを次のように設定する必要があります。

```
NLS_LANGUAGE = FRENCH
```

この場合、次のサーバー・メッセージ、

```
ORA-00942: table or view does not exist
```

は、次のように表示されます。

```
ORA-00942: table ou vue inexistante
```

サーバーが使用するメッセージは、ORA_RDBMS ディレクトリにあるバイナリ形式ファイル、またはそれに相当するファイルに格納されます。このファイルは、ファイル名規則に従って、サポートされる言語ごとに 1 バージョンずつ、複数のバージョンを持つことができます。

<*product_id*><*language_abbrev*>.MSB

たとえば、フランス語のサーバー・メッセージを格納するファイルは、ORAF.MSB と呼ばれ、"F" は、フランス語の略称です。

メッセージは、特定のマシンおよびオペレーティング・システムに応じて、特定のキャラクタ・セットでこれらのファイルに格納されます。このキャラクタ・セットがデータベースのキャラクタ・セットと異なる場合は、メッセージ・テキストは、データベースのキャラクタ・セットに自動的に変換されます。クライアントのキャラクタ・セットがデータベースのキャラクタ・セットと異なる場合は、メッセージ・テキストは、必要に応じてさらにクライアントのキャラクタ・セットに変換されます。メッセージは、このようにキャラクタ・セット規則の制限によって、ユーザーの端末に正しく表示されます。

NLS_LANGUAGE のデフォルト値は、オペレーティング・システム固有の値にできます。初期化ファイル内の値を変更してからインスタンスを再起動することで、NLS_LANGUAGE パラメータを変更できます。

デフォルト値の詳細は、オペレーティング・システム固有の Oracle ドキュメントを参照してください。

次に、NLS_LANGUAGE を設定する前と後の動作例を示します。

```
SVRMGR> ALTER SESSION SET NLS_LANGUAGE Italian
SVRMGR> SELECT ename, hiredate, ROUND(sal/12,2) sal FROM emp;
ENAME     HIREDATE    SAL
-----     --------    ---
Clark     09-Dic-88   4195.83
Miller    23-Mar-87   4366.67
Strauß    01-Apr-95   3795.87

SVRMGR> ALTER SESSION SET NLS_LANGUAGE German
SVRMGR> SELECT ename, hiredate, ROUND(sal/12,2) sal FROM emp;
ENAME     HIREDATE    SAL
-----     --------    ---
Clark     09-DEZ-88   4195.83
Miller    23-MÄR-87   4366.67
Strauß    01-APR-95   3795.87
```

NLS_TERRITORY

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータおよび ALTER SESSION
デフォルト値 : NLS_LANG
値の範囲 : 有効な地域名

NLS_TERRITORY は、次のデフォルトの日付および数値書式化の特性に関する規則を指定します。

- 日付書式
- 小数点文字およびグループ・セパレータ
- 各国通貨記号
- ISO 通貨記号
- 第二通貨記号
- 週の最初の曜日
- 貸方および借方記号
- ISO 週フラグ
- リスト・セパレータ

初期化ファイル内の NLS_TERRITORY に指定した値は、インスタンスのデフォルト値です。たとえば、デフォルトをフランスに指定するには、このパラメータを次のように設定する必要があります。

```
NLS_TERRITORY = FRANCE
```

この場合、数値は、カンマを小数点文字として使用して書式化されます。

初期化ファイル内の値を変更してからインスタンスを再起動することで、NLS_TERRITORY パラメータを変更できます。NLS_TERRITORY のデフォルト値は、オペレーティング・システム固有の値にできます。

次に、NLS_TERRITORY を設定する前と後の動作例を示します。

```
SQL> describe SalaryTable;
Name                          Null?          TYPE
---------                     -------        ------
SALARY                                       NUMBER
```

```
SQL> column SALARY format L999,999.99;
SQL> SELECT * from SalaryTable;
               SALARY
---------------------
          $100,000.00
          $150,000.00

SQL> ALTER SESSION SET NLS_TERRITORY = Germany;
Session altered.

SQL> SELECT * from SalaryTable;
               SALARY
---------------------
         DM100,000.00
         DM150,000.00

SQL> ALTER SESSION SET NLS_LANGUAGE = German;
Sitzung wurde geandert.

SQL> SELECT * from SalaryTable;
               SALARY
---------------------
         DM100,000.00
         DM150,000.00

SQL> ALTER SESSION SET NLS_TERRITORY = France;
Sitzung wurde geandert.

SQL> SELECT * from SalaryTable;
               SALARY
---------------------
          F100,000.00
          F150,000.00
```

通貨単位の記号は変更されていますが、通貨変換の計算は行われていません。

ALTER SESSION

デフォルトの言語および地域は、ALTER SESSION 文を使用してセッション中に変更できます。たとえば、次のように指定します。

```
% setenv NLS_LANG Italian_Italy.WE8DEC

SVRMGR> SELECT ename, hiredate, ROUND(sal/12,2) sal FROM emp;
ENAME     HIREDATE    SAL
-----     --------    ---
```

```
Clark     09-Dic-88   4195,83
Miller    23-Mar-87   4366,67
Strauß    01-Apr-95   3795,87

SVRMGR> ALTER SESSION SET NLS_LANGUAGE = German
2     > NLS_DATE_FORMAT = 'DD.MON.YY'
3     > NLS_NUMERIC_CHARACTERS = '.,';

SVRMGR> SELECT ename, hiredate, ROUND(sal/12,2) sal FROM emp;
ENAME     HIREDATE    SAL
-----     --------    ---
Clark     09.DEZ.88   4195.83
Miller    23.MÄR.87   4366.67
Strauß    01.APR.95   3795.87
```

この機能によって、それぞれのセッションについてのデータベースの言語環境が自動的に決定されます。セッションがデータベースに接続し、データベース・パラメータの NLS_LANGUAGE および NLS_TERRITORY の値を、NLS_LANG の *language* 引数および *territory* 引数で指定された値に設定すると、ALTER SESSION 文が自動的に実行されます。NLS_LANG が定義されていないと、暗黙の ALTER SESSION 文は実行されません。

NLS_LANG が定義されていると、セッションが接続（直接接続と間接接続の両方）するインスタンスすべてに対して暗黙の ALTER SESSION が実行されます。NLS パラメータの値が、ALTER SESSION によってセッション中に明示的に変更されると、その変更は、そのユーザー・セッションが接続しているすべてのインスタンスに反映されます。

メッセージおよびテキスト

メッセージおよびテキストはすべて同じ言語でなければなりません。たとえば、Developer 2000 アプリケーションの実行中に表示されるメッセージおよびボイラープレート・テキストは、次の 3 つが元になっています。

- サーバーからのメッセージ
- SQL*Forms によって生成されたメッセージおよびボイラープレート・テキスト
- アプリケーションの一部として定義されたメッセージおよびボイラープレート・テキスト

3 番目の要件は、アプリケーションにて処理する必要があります。NLS では、残りの 2 つを処理します。

時刻パラメータ

世界中で多種多様な時刻書式が使用されています。次に、標準の時刻書式をいくつか示します。

国名	説明	例
フィンランド	hh24:mi:ss	13:50:23
ドイツ	hh24:mi:ss	13:50:23
日本	hh24:mi:ss	13:50:23
UK	hh24:mi:ss	13:50:23
US	hh:mi:ss am	1.50.23 PM

日付パラメータ

Oracle では、日付パラメータを使用して日付がどのように表示されるかを制御できます。

日付書式

世界中で多種多様な日付書式が使用されています。次に、代表的な日付書式をいくつか示します。

国名	説明	例
フィンランド	dd.mm.yyyy	28.02.1998
ドイツ	dd.mm.yy	28.02.98
日本	yy-mm-dd	98-02-28
UK	dd-mon-yy	28-Feb-98
US	dd-mon-yy	28-Feb-98

NLS_DATE_FORMAT

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 : 特定の地域のデフォルト書式
値の範囲 : 有効な日付書式マスク

このパラメータは、TO_CHAR 関数および TO_DATE 関数で使用するデフォルトの日付書式を定義します。このパラメータのデフォルト値は、NLS_TERRITORY によって決定されます。このパラメータの値は、任意の有効な日付書式マスクにすることができ、その値は引用符で囲む必要があります。たとえば、次のように指定します。

```
NLS_DATE_FORMAT = "MM/DD/YYYY"
```

日付書式に文字列リテラルを追加する場合は、その文字列リテラルを二重引用符で囲みます。二重引用符などの特殊文字を使用する場合は、必ずエスケープ文字を前に付けてください。また、式全体は引用符で囲んでください。たとえば、次のように指定します。

```
NLS_DATE_FORMAT = '\"Today\'s date\" MM/DD/YYYY'
```

別の例では、デフォルト日付書式を設定して、月をローマ数字で表示するために、初期化ファイルに次の行を加えています。

```
NLS_DATE_FORMAT = "DD RM YYYY"
```

このようなデフォルトの日付書式を使用すると、次の SELECT 文で、ローマ数字を使用した月が戻ります（現在の日付を 1997 年 2 月 12 日とした場合）。

```
SELECT TO_CHAR(SYSDATE) CURRDATE
     FROM DUAL;
CURRDATE
---------
12 II 1997
```

このパラメータの値は、内部日付書式で格納されます。それぞれの書式要素は 2 バイトを占め、文字列はそれぞれ、文字列のバイト数に 1 つの終了文字のバイト数を加えたバイト数になります。また、書式マスク全体で 2 バイトの終了文字があります。たとえば、"MM/DD/YY" は、内部で 12 バイトを占めます。これは、書式マスクが 3 つの書式要素、2 つの 1 バイト文字列（2 個のスラッシュ）、および 2 バイトの終了文字で構成されるためです。このパラメータ値の書式は、24 バイト以内で設定しなくてはなりません。

注意 : アプリケーションを設計する場合、可変長のデフォルト日付書式を受け入れなければならないことがあります。また、パラメータ値は二重引用符で囲まなければなりません。引用符は、書式マスクの一部として解釈されます。

初期化ファイル内の NLS_DATE_FORMAT の値を変更してからインスタンスを再起動することによって、そのデフォルト値を変更できます。また、ALTER SESSION SET NLS_DATE_FORMAT コマンドを使用して、セッション中にその値を変更できます。

西暦 2000 年問題

現在、ほとんどの地域のデフォルト日付書式では、年の書式として西暦年の下 2 桁を示す "YY" を指定しています。アプリケーションが 2000 年に対応している場合、"YYYY" または "RRRR" を使用して NLS_DATE_FORMAT を指定しても問題ありません。アプリケーションが 2000 年に対応していない場合は、NLS_DATE_FORMAT に "RR" を指定する必要がある場合があります。"RR" 書式には、次の効果があります : 年を下 2 桁で指定すると、指定した年が 50 より小さくて現在の西暦年の下 2 桁が 50 以上である場合、RR は次の世紀で年の値を返します。また、指定した年が 50 以下で現在の西暦年の下 2 桁が 50 より小さい場合、RR は前の世紀で年の値を返します。

日付書式要素の詳細は、『Oracle8*i* SQL リファレンス』の「日付書式モデル」を参照してください。

日付書式およびパーティション・バウンド式

日付列のパーティション・バウンド式には、月、日および 4 桁の西暦を指定する必要がある書式を使用して日付を指定しなければなりません。たとえば、日付書式 MM-DD-YYYY には、月、日および 4 桁の西暦をすべて指定する必要があります。日付書式 DD-MON-YY（11-jan-97 など）は無効です。この書式では現在の世紀の日付しか表示できないためです。

TO_DATE() を使用して、月、日および 4 桁の西暦の指定が要求される日付書式を指定します。たとえば、次のように指定します。

```
TO_DATE('11-jan-1997', 'dd-mon-yyyy')
```

NLS_DATE_FORMAT で指定するセッションのデフォルトの日付書式に、現在の世紀以外の日付を指定できない場合（つまり、デフォルトの日付書式が MM-DD-YY の場合）は、次のいずれかを行ってください。

- TO_DATE() を使用して、月、日および 4 桁の西暦の指定が要求される書式で日付を指定します。
- セッションに指定した NLS_DATE_FORMAT の値を変更して、月、日および 4 桁の西暦の指定が要求される書式で日付を指定できるようにします。

TO_DATE() の使用方法の詳細は、『Oracle8*i* SQL リファレンス』を参照してください。

NLS_DATE_LANGUAGE

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 : 日付に使用されるデフォルトの言語
値の範囲 : 有効な言語名

このパラメータは、TO_CHAR 関数および TO_DATE 関数によって曜日名および月名の表記に使用する言語を指定し、NLS_LANGUAGE で暗黙に指定された言語を変更します。NLS_DATE_LANGUAGE の構文は、NLS_LANGUAGE パラメータと同じで、サポートされているすべての言語が有効な値です。たとえば、日付の言語をフランス語に指定するには、このパラメータを次のように設定する必要があります。

```
NLS_DATE_LANGUAGE = FRENCH
```

この場合、問合せは次のようになります。

```
SELECT TO_CHAR(SYSDATE, 'Day:Dd Month yyyy')
     FROM DUAL;
```

この問合せによって、次の結果が戻ります。

```
Mercredi:12 Février 1997
```

月および曜日の略称も、指定した言語で表示されます。たとえば、次のとおりです。

```
Me:12 Fév 1997
```

また、デフォルトの日付書式では、言語固有の月の略称を使用します。たとえば、デフォルトの日付書式が DD-MON-YYYY である場合、前述の日付は次のように挿入されます。

```
INSERT INTO tablename VALUES ('12-Fév-1997');
```

AM、PM、AD および BC の略称も NLS_DATE_LANGUAGE によって指定した言語で戻ります。TO_CHAR 関数を使用して表記される数字には、常に英語が使用されます。たとえば、次のとおりです。

```
SELECT TO_CHAR(TO_DATE('12-Fév'),'Day: ddspth Month')
     FROM DUAL;
```

これによって、次の結果が戻ります。

```
Mercredi: twenty-seventh Février
```

初期化ファイル内の NLS_DATE_LANGUAGE の値を変更してからインスタンスを再起動することによって、そのデフォルト値を変更できます。また、ALTER SESSION SET NLS_DATE_LANGUAGE コマンドを使用して、セッション中にその値を変更できます。

カレンダ・パラメータ

Oracle では、パラメータを使用してカレンダに関する項目を制御できます。

カレンダ書式

地域ごとに格納されるカレンダ情報の種類は、次のとおりです。

- 週の最初の曜日
- 年の最初の暦週
- 1 年の日数および月数
- 時代の最初の年

週の最初の曜日

文化によっては、日曜日を週の最初の曜日と考えます。また、月曜日を週の最初の曜日と考える文化もあります。ドイツのカレンダは、月曜日から始まります。

			März	**1998**		
Mo	Di	Mi	Do	Fr	Sa	So
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

年の最初の暦週

ドイツなどの多くの国では、週間を使用してスケジューリング、計画、および経理を行います。Oracle では、この規則をサポートします。

ISO 規格では、ISO 週番号に関連する年は、カレンダ年と異なることがあります。たとえば、1988 年 1 月 1 日は、1987 年の ISO 週番号 53 になります。週は、常に月曜日に始まって日曜日に終わります。

- 1 月 1 日が金曜日、土曜日または日曜日であれば、1 月 1 日を含む週は、この週の大半の日が前年に属するため、前年の最後の週になります。
- 1 月 1 日が月曜日、火曜日、水曜日または木曜日であれば、1 月 1 日を含む週は、この週の大半の日が新年に属するため、新年の最初の週になります。

ISO 規格をサポートするために、ISO 週番号を戻す書式要素 IW があります。

次に、最初の週が 4 日間以上である一般的な例を示します。

		January		**1998**			
Mo	Tu	We	Th	Fr	Sa	Su	
			1	2	3	4	<= 1998 年の第 1 週
5	6	7	8	9	10	11	<= 1998 年の第 2 週
12	13	14	15	16	17	18	<= 1998 年の第 3 週
19	20	21	22	23	24	25	<= 1998 年の第 4 週
26	27	28	29	30	31		<= 1998 年の第 5 週

次に、最初の週が 3 日間以下である一般的な例を示します。

		January		**1999**			
Mo	Tu	We	Th	Fr	Sa	Su	
				1	2	3	<= 1998 年の第 53 週
4	5	6	7	8	9	10	<= 1999 年の第 1 週
11	12	13	14	15	16	17	<= 1999 年の第 2 週
18	19	20	21	22	23	24	<= 1999 年の第 3 週
25	26	27	28	29	30	31	<= 1999 年の第 4 週

1 年の日数および月数

Oracle では、デフォルトのグレゴリオ暦だけでなく、他に 6 つの暦法をサポートしています。

- Japanese Imperial（元号）— グレゴリオ暦と同じ月数および日数を使用しますが、年は各元号ごとに始まります。
- ROC Official（台湾の公式の暦）— グレゴリオ暦と同じ月数および日数を使用しますが、年は台湾の建国から始まります。
- Persian（ペルシャ暦）—12 か月間あり、それぞれの月は同じ長さです。
- Thai Buddha（タイ仏教暦）— 仏教のカレンダを使用します。
- Arabic Hijrah（イスラム紀元）—12 か月間あり、日数は 354 または 355 です。
- English Hijrah（英語版イスラム紀元）—12 か月間あり、日数は 354 または 355 です。

時代の最初の年

イスラム暦は、ヒジュラ紀元の年から始まります。元号暦は、天皇が在位した最初の年から始まります。たとえば、1998 年は平成 10 年になります。しかし、グレゴリア暦も広く理解されているので、1998 を表現するために 98 と 10 の両方が使用されます。

NLS_CALENDAR

パラメータ・タイプ :	文字列
パラメータの有効範囲 :	初期化パラメータおよび ALTER SESSION
デフォルト値 :	Gregorian（グレゴリオ暦）
値の範囲 :	有効なカレンダの書式名

世界中で多種多様な暦法が使用されています。NLS_CALENDAR によって、Oracle が使用する暦法を指定できます。

このパラメータは、次の値のいずれかをとることができます。

- Arabic Hijrah（イスラム紀元）
- English Hijrah（英語版イスラム紀元）

- Gregorian（グレゴリオ暦）
- Japanese Imperial（日本の元号制）
- Persian（ペルシャ暦）
- ROC Official（台湾の公式の暦）
- Thai Buddha（タイ仏教暦）

たとえば、NLS_CALENDAR が「Japanese Imperial」に設定され、日付書式が「E YY-MM-DD」であり、日付が 1997 年 5 月 15 日の場合、SYSDATE は次のように表示されます。

```
SELECT SYSDATE FROM DUAL;
SYSDATE
--------
H 09-05-15
```

数値パラメータ

Oracle では、どのように数字が表示されるかを制御できます。

数値書式

データベースでは、数字列を正確に解釈するために、それぞれのセッションで使用される数値書式化の規則の認識が必要です。たとえば、数値の入力時に小数点文字としてピリオドまたはカンマのどちらを使用するか（234.00 または 234,00）などがデータベースで認識されている必要があります。同じように、アプリケーションでは、クライアント側で想定している書式で数値情報を表示できるように設定されている必要があります。

次に、一般的な数値書式をいくつか示します。

国名	例
フィンランド	1.234.567,89
ドイツ	1.234.567,89
日本	1,234,567.89
UK	1,234,567.89
US	1,234,567.89

NLS_NUMERIC_CHARACTERS

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 : 小数点文字およびグループ・セパレータ
値の範囲 : 有効な数値書式マスク

このパラメータは、小数点文字およびグループ・セパレータを指定して、NLS_TERRITORY で暗黙に定義された小数点文字およびグループ・セパレータを変更します。グループ・セパレータとは、整数グループ（つまり、千、100 万、10 億など）を区切る文字です。小数点文字は、数値の整数部と小数部を区切ります。

任意の文字を、小数点またはグループ・セパレータにできます。指定する 2 つの文字は、シングルバイトであること、および両方の文字が互いに異なっていることが必要です。これらの文字には、数字、プラス（+）、ハイフン（-）、小なり記号（<）、または大なり記号（>）は使用できません。

文字は次の書式で指定します。

```
NLS_NUMERIC_CHARACTERS = "<decimal_character><group_separator>"
```

グループ・セパレータは、数値書式マスク G によって戻される文字です。たとえば、小数点文字をカンマに設定し、グループ・セパレータをピリオドに設定するには、このパラメータを次のように設定する必要があります。

```
NLS_NUMERIC_CHARACTERS = ",."
```

文字は両方シングルバイトで、互いに異なっていなければなりません。どちらかを空白にすることもできます。

注意 : 小数点文字がピリオド（.）以外の場合、またはグループ・セパレータを使用する場合は、SQL 文に指定する数字は引用符で囲まなければなりません。たとえば、前述の NLS_NUMERIC_CHARACTERS の値では、次の SQL 文の数値リテラルは引用符で囲む必要があります。

```
INSERT INTO SIZES (ITEMID, WIDTH, QUANTITY)
     VALUES (618, '45,5', TO_NUMBER('1.234','9G999'));
```

次のいずれかの方法で NLS_NUMERIC_CHARACTERS のデフォルト値を変更できます。

- 初期化ファイルで NLS_NUMERIC_CHARACTERS の値を変更し、インスタンスを再起動する。

- ALTER SESSION SET NLS_NUMERIC_CHARACTERS コマンドを使用して、セッション中の NLS_NUMERIC_CHARACTERS パラメータの値を変更する。

通貨パラメータ

Oracle では、通貨記号および財務記号がどのように表示されるかを制御できます。

通貨書式

世界中で多種多様な通貨書式が使用されています。次に、一般的な通貨書式をいくつか示します。

国名	例
フィンランド	1.234,56 mk
ドイツ	1.234,56 DM
日本	¥1,234.56
UK	£1,234.56
US	$1,234.56

NLS_CURRENCY

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 : 各国通貨記号
値の範囲 : 有効な書式名

このパラメータは、数値書式マスク L によって戻される文字列である各国通貨記号を指定し、NLS_TERRITORY で暗黙に定義された各国通貨記号を変更します。たとえば、各国通貨記号を「Dfl」（空白を含む）に設定するには、このパラメータを次のように設定する必要があります。

```
NLS_CURRENCY = "Dfl "
```

この場合、問合せは次のようになります。

```
SELECT TO_CHAR(TOTAL, 'L099G999D99') "TOTAL"
     FROM ORDERS WHERE CUSTNO = 586
```

この問合せによって、次の結果が戻ります。

```
TOTAL
-------------
Dfl 12.673,49
```

初期化ファイル内の NLS_CURRENCY の値を変更してからインスタンスを再起動することによって、そのデフォルト値を変更できます。また、ALTER SESSION SET NLS_CURRENCY コマンドを使用して、セッション中にその値を変更できます。

NLS_ISO_CURRENCY

パラメータ・タイプ :	文字列
パラメータの有効範囲 :	初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 :	ISO 国際通貨記号
値の範囲 :	有効な地域名

このパラメータは、数値書式マスク C によって戻される文字列（ISO 通貨記号）を指定します。この結果、NLS_TERRITORY で暗黙に定義されている ISO 通貨記号は無効になります。

各国通貨記号は不明確な場合があります。たとえば、ドル記号（$）は米国ドルを指すこともオーストラリア・ドルを指すこともあります。ISO 仕様 4217 1987-07-15 では、特定の地域（または国）の通貨について固有の国際通貨記号が定義されています。

たとえば、米国ドルに対する ISO 通貨記号は USD であり、オーストラリア・ドルの場合は AUD です。ISO 通貨記号を指定するために、それに対応する地域名が使用されます。

NLS_ISO_CURRENCY の構文は、NLS_TERRITORY パラメータと同じで、サポートされているすべての地域が有効な値です。たとえば、フランスに対する ISO 通貨記号を指定するには、このパラメータを次のように設定する必要があります。

```
NLS_ISO_CURRENCY = FRANCE
```

この場合、問合せは次のようになります。

```
SELECT TO_CHAR(TOTAL, 'C099G999D99') "TOTAL"
     FROM ORDERS WHERE CUSTNO = 586
```

この問合せによって、次の結果が戻ります。

```
TOTAL
-------------
FRF12.673,49
```

初期化ファイル内の NLS_ISO_CURRENCY の値を変更してからインスタンスを再起動することによって、そのデフォルト値を変更できます。また、ALTER SESSION SET NLS_ISO_CURRENCY コマンドを使用して、セッション中にその値を変更できます。

一般的な ISO 通貨記号は次のとおりです。

国名	例
フィンランド	1.234.567,89 FIM
ドイツ	1.234.567,89 DEM
日本	1,234,567.89 JPY
UK	1,234,567.89 GBP
US	1,234,567.89 USD

NLS_DUAL_CURRENCY

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 : 第二通貨記号
値の範囲 : 有効な名前

このパラメータは、地域で指定されている第二通貨記号を変更します。NLS_DUAL_CURRENCY を設定せずに新しいセッションを開始すると、現行の言語環境地域に定義されているデフォルトの第二通貨記号が使用されます。NLS_DUAL_CURRENCY を設定すると、その値を第二通貨記号に使用してセッションを開始します。

NLS_DUAL_CURRENCY は、ユーロ（欧州統一通貨）をサポートするために導入されました。ユーロ記号をサポートしているキャラクタ・セットは、次のとおりです。

表 2-3 ユーロ記号をサポートしているキャラクタ・セット

名前	説明	ユーロのコード値
D8EBCDIC1141	EBCDIC コード・ページ 1141 8 ビット・オーストリア・ドイツ語	0x9F
DK8EBCDIC1142	EBCDIC コード・ページ 1142 8 ビット・デンマーク語	0x5A
S8EBCDIC1142	EBCDIC コード・ページ 1143 8 ビット・スウェーデン語	0x5A
I8EBCDIC1144	EBCDIC コード・ページ 1144 8 ビット・イタリア語	0x9F

表 2-3 ユーロ記号をサポートしているキャラクタ・セット

名前	説明	ユーロのコード値
F8EBCDIC1147	EBCDIC コード・ページ 1147 8 ビット・フランス語	0x9F
WE8PC858	IBM-PC コード・ページ 858 8 ビット西欧	0xD5
WE8ISO8859P15	ISO 8859-15 西欧	0xA4
EE8MSWIN1250	MS Windows コード・ページ 1250 8 ビット東欧	0x80
CL8MSWIN1251	MS Windows コード・ページ 1251 8 ビット・ラテン語 / キリル文字	0x88
WE8MSWIN1252	MS Windows コード・ページ 1252 8 ビット西欧	0x80
EL8MSWIN1253	MS Windows コード・ページ 1253 8 ビット・ラテン語 / ギリシャ語	0x80
TR8MSWIN1254	MS Windows コード・ページ 1254 8 ビット・トルコ語	0x80
BLT8MSWIN1257	MS Windows コード・ページ 1257 バルト語	0x80
VN8MSWIN1258	MS Windows コード・ページ 1253 8 ビット・ベトナム語	0xA0
TH8TISASCII	タイ工業規格 520-2533 - ASCII 8 ビット	0x80
AL24UTFFSS	Unicode 1.1 UTF-8 ユニバーサル・キャラクタ・セット	U+20AC
UTF8	Unicode 2.0 UTF-8 ユニバーサル・キャラクタ・セット	U+20AC

NLS_MONETARY_CHARACTERS

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 : NLS_TERRITORY から導出される
値の範囲 : 有効な名前

NLS_MONETARY_CHARACTERS は、米国ドルを表すドル記号（$）およびセントを表すセント記号（¢）などの通貨単位を示す文字を指定します。

指定する 2 つの文字はシングルバイトにする必要があり、互いに同じ文字は指定できません。また、この 2 つの文字には、数字、プラス（+）、ハイフン（-）、小なり記号（<）または大なり記号（>）は使用できません。

NLS_CREDIT

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 : NLS_TERRITORY から導出される
値の範囲 : 最大 9 バイト（NULL を含まない）の任意の文字列

NLS_CREDIT は、財務レポートで貸方を表示する記号を設定します。このパラメータのデフォルト値は、NLS_TERRITORY によって決定されます。

NLS_DEBIT

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 : NLS_TERRITORY から導出される
値の範囲 : 最大 9 バイト（NULL を含まない）の任意の文字列

NLS_DEBIT は、財務レポートで借方を表示する記号を設定します。このパラメータのデフォルト値は、NLS_TERRITORY によって決定されます。

照合パラメータ

Oracle では、照合パラメータを使用して日付をどのようにソートするかを選択できます。

ソート順

ソート順は言語によって異なります。さらに、異なる文化または国が同じアルファベットを使用していたとしても、ソート・ワードが異なる場合があります。たとえば、ドイツとオーストリアでは、ドイツ語の強い発音の s を表すエスツェット（ß）のソート結果は異なります。ドイツ語の言語ソート基準では、この順序を *SS* の 2 文字としてソートし、一方、オーストリアの言語ソート基準では、これを *SZ* としてソートします。もう 1 つの例は、*ö*、*o* および *œ* の扱いです。これらは、さまざまなゲルマン語を使用してソートされます。

Oracle では、多種多様なソート・タイプを使用できますが、言語的に正確なソートを実現すると、パフォーマンスが低下する場合があります。どちらを優先するかは、データベース管理者がその場その場の基準に従って決定してください。典型的な事例として、スペイン語をソートする場合があります。伝統的なスペイン語では、*ch* および *ll* は特有の文字です。したがって、正しいソート順は、*cerveza*、*colorado*、*cheremoya*、*lago*、*luna*、*llama* となります。しかし、正確な言語ソートによってパフォーマンスの低下が生じます。

東アジア各国語のソートは、困難で複雑です。現在 Oracle では、特定のキャラクタ・セットのバイナリ順序に従って、東アジア各国語をソートしています。たとえば、シフト JIS のキャラクタ・セット表は漢字の部首によって順序付けられているので、シフト JIS 環境でソートする場合は、そのバイナリ順序が使用されます。

文字データのソート

従来、文字データがソートされる場合は、そのソート基準は、文字コード体系によって定義された文字の数値に基づいています。このようなソートをバイナリ・ソートと呼びます。ASCII 規格および EBCDIC 規格では、文字 A ～ Z を昇順の数値で定義しているため、英語のアルファベットについては正しいソート結果が得られます。

ただし、ASCII 規格では、大文字はすべて小文字の前にきます。逆に、EBCDIC 規格では、小文字はすべて大文字の前にきます。

バイナリ・ソート

その他の言語で使用されている文字が存在すると、一般に、バイナリ・ソートでは正しい結果が得られません。たとえば、文字コード体系で Ä が B よりも高い数値を持っている場合、昇順の ORDER BY 問合せでは ABC、ABZ、BCD、ÄBC の順で文字列が戻ります。

言語ソート

特定の言語に準じた文字の順序（アルファベット順など）に一致するソート基準を得るためには、文字コード体系内の数値に依存せずに文字をソートする別のソート技法が必要です。この技法を言語ソートと呼びます。バイナリ・ソートで希望する結果が戻るように、言語的に適切な文字順序を反映する別の 2 進値に文字を置換することによって、言語ソートは動作します。

Oracle Server では、前述の 2 つのソート方法を提供します。言語ソート基準は、言語依存データの一部として定義されます。それぞれの言語ソート基準には、一意の名前があります。NLS パラメータによって、ORDER BY 問合せのソート方法を定義します。デフォルト値を指定でき、この値はセッションごとに NLS_SORT パラメータで変更できます。言語の定義の全リストは、A-17 ページ「言語の定義」に示されています。

警告 : アジア地域のマルチバイト・キャラクタ・セットでは、言語ソートをサポートしていません。データベース・キャラクタ・セットがマルチバイトの場合は、バイナリ・ソートが使用できます。このバイナリ・ソートでは、ソート基準をキャラクタ・セットの仕様に基づくようにします。日本語のひらがな / カタカナのキャラクタ・セット、および UTF8 のキャラクタ・セットは例外です。漢字の読みにあたるひらがなまたはカタカナを使用して別の列を作成し、その列でソートすることによってのみ、日本語の読みでのソートが実現できます。

言語索引

英語以外の言語を使用するファンクション・ベースの索引を作成できます。次に、簡単な例を示します。

```
SVRMGR> CREATE INDEX nls_index ON my_table (NLSSORT(name, 'NLS_SORT = German'));
```

次の文を入力します。

```
SVRMGR> SELECT * FROM my_table ORDER BY name;
```

ドイツ語の照合基準に従って、列が戻されます。

詳細は、『Oracle8*i* 概要』のファンクション・ベースの索引に関する説明を参照してください。

大文字と小文字を区別しないソート

大文字と小文字を区別せずに検索できるようにするファンクション・ベースの索引を作成できます。たとえば、次のように指定します。

```
SVRMGR> CREATE INDEX case_insensitive_ind ON my_table(NLS_UPPER(empname));
SVRMGR> SELECT * FROM my_table WHERE NLS_UPPER(empname) = 'KARL';
```

詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』のファンクション・ベースの索引に関する説明を参照してください。

言語的に特殊な事例

言語上の特殊な事例として、ソート時に連続した複数の文字を 1 つの文字として扱わなければならない場合があります。このような特殊な事例は、言語ソートを使用すると、自動的に処理されます。たとえば、スペイン語の言語ソート基準の 1 つでは、二重文字の *ch* および *ll* は、それぞれ *c* と *d* の間および *l* と *m* の間の単一文字としてソートされるよう指定します。

もう 1 つの例は、ドイツ語の強い発音の s を表すエスツェット（ß）です。ドイツ語の言語ソート基準では、この順序を *SS* の 2 文字としてソートし、一方、オーストリアの言語ソート基準では、これを *SZ* としてソートします。

このような特殊事例の処理は、大文字を小文字に（またはその逆に）変換するときにも実行されます。たとえば、ドイツ語では、強い発音の s を表すエスツェット（ß）の大文字は SS の 2 文字になります。このような大 / 小文字変換の問題は、言語ソート基準で設定された規則に従って、NLS_UPPER、NLS_LOWER および NLS_INITCAP 関数によって処理されます。（標準関数 UPPER、LOWER および INITCAP では、これらの特殊事例は処理されません。）

NLS_SORT

パラメータ・タイプ :	文字列
パラメータの有効範囲 :	初期化パラメータ、環境変数および ALTER SESSION
デフォルト値 :	文字のソート基準
値の範囲 :	BINARY または有効な言語定義名

このパラメータは、文字データのソートのタイプを指定します。その結果、NLS_LANGUAGE で暗黙に定義されたソートのタイプは無効になります。

NLS_SORT の構文は次のとおりです。

```
NLS_SORT = { BINARY | name }
```

BINARY はバイナリ・ソートを指定し、*name* は特定の言語ソート基準を指定します。たとえば、German という言語ソート基準を指定するには、このパラメータを次のように設定する必要があります。

```
NLS_SORT = German
```

言語ソート基準に付けられる名前は、言語名と直接的な関係はありません。ただし、通常は、サポートされている各言語には、言語名と同じ名前を使用する適切な言語ソート基準が定義されます。

注意 : NLS_SORT パラメータが BINARY に設定されていると、オプティマイザは索引スキャンを選択することでソートを行わずに ORDER BY 句を実行できる場合があります。一方、NLS_SORT が言語ソートに設定されている場合は、ORDER BY 句を実行するには必ずソートが必要になります。

初期化ファイル内の NLS_SORT の値を変更してからインスタンスを再起動することによって、そのデフォルト値を変更できます。また、ALTER SESSION SET NLS_SORT コマンドを使用して、セッション中にその値を変更できます。

言語の定義の全リストは、表 A-8「言語の定義」に示されています。

NLS_COMP

パラメータ・タイプ :	文字列
パラメータの有効範囲 :	環境変数および ALTER SESSION
デフォルト値 :	BINARY
値の範囲 :	BINARY または ANSI

このパラメータを使用すると、SQL 文中に NLS_SORT を使用する面倒な処理を回避できます。通常、WHERE 句での比較はバイナリで行われます。言語上の比較を実行するには、NLSSORT 関数を使用しなければなりません。NLSSORT 関数の使用は、必要な言語ソートがすでに NLS_SORT セッションのパラメータに指定されている場合など、特に面倒に感じられます。このような場合、NLS_COMP を使用すると、比較が NLS_SORT セッション・パラメータに従って言語的に行われるように指示できます。セッションの変更で次のように入力します。

```
SVRMGR> ALTER SESSION SET NLS_COMP = ANSI;
```

WHERE 句での比較が必ずバイナリで行われるように指定するには、次のように入力します。

```
SVRMGR> ALTER SESSION SET NLS_COMP = BINARY;
```

NLS_COMP に ANSI が設定されている場合、言語上の順序が希望されている列には言語索引が存在しなければなりません。

言語索引を使用可能にするには、次の構文を使用します。

```
SVRMGR> CREATE INDEX i ON t(NLSSORT(col, 'NLSSORT=FRENCH'));
```

NLS_LIST_SEPARATOR

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 初期化パラメータおよび ALTER SESSION
デフォルト値 : NLS_TERRITORY から導出される
値の範囲 : 有効な任意の文字

NLS_LIST_SEPARATOR は、列記された値を区切るために使用する文字を指定します。

指定する文字には、シングルバイトにする必要があり、数値の小数点文字、通貨の小数点文字、数字、プラス（+）、ハイフン（-）、小なり記号（<）、大なり記号（>）、またはピリオド（.）のいずれも使用できません。

キャラクタ・セット・パラメータ

Oracle では、クライアントで使用するキャラクタ・セットを指定できます。

NLS_NCHAR

パラメータ・タイプ : 文字列
パラメータの有効範囲 : 環境変数
デフォルト値 : NLS_LANG から導出される
値の範囲 : 有効な任意のキャラクタ・セット名

NLS_NCHAR は、各国キャラクタ・セットとしてクライアント・アプリケーションで使用するキャラクタ・セットを指定します。これを指定しないと、クライアント・アプリケーションはデータベース・キャラクタ・セット・データと同じキャラクタ・セットを使用します。

3

キャラクタ・セットの選択

この章では、キャラクタ・セットを選択するときに理解しておく必要がある NLS トピックについて説明します。トピックは次のとおりです。

コード化キャラクタ・セットとは

キャラクタ・セットはデータベースを作成する場合に指定します。キャラクタ・セットの選択によって、データベース内で表現される言語が決定します。この選択は、データベース・スキーマの作成、および文字データを処理するアプリケーションの開発に影響を及ぼします。また、オペレーティング・システムのリソースとデータベースのパフォーマンスの相互運用にも影響します。

文字を処理する場合、コンピュータ・システムは文字データをグラフィカルな表現としてではなく数値コードとして処理します。たとえば、データベースに文字「A」を格納すると、実際はソフトウェアによって解釈される数値コードが格納されます。

文字の集まり（アルファベット文字、表意文字、記号、句読点、制御文字など）は、コード化キャラクタ・セットとしてコード化できます。コード化キャラクタ・セットでは、文字レパートリ内のそれぞれの文字に一意の数値コードが割り当てられています。次に、数値コードが割り当てられている文字の例を示します。

表 3-1 ASCII キャラクタ・セットでのコード化文字

文字	説明	コード値
!	感嘆符	0x21
#	シャープ記号	0x23
$	ドル記号	0x24
1	数字の 1	0x31
2	数字の 2	0x32
3	数字の 3	0x33
A	大文字の A	0x41
B	大文字の B	0x42
C	大文字の C	0x43
a	小文字の a	0x61
b	小文字の b	0x62
c	小文字の c	0x63

多種多様なコード化キャラクタ・セットが、コンピュータ業界で使用され、Oracle によってサポートされています。Oracle では、各国の、国際的なおよびベンダー固有のコード化キャラクタ・セット規格のほとんどがサポートされています。Oracle がサポートしているキャラクタ・セットの全リストは、付録 A「ロケール・データ」に示されています。キャラクタ・セットは、次の点で異なります。

- 使用可能な文字数
- 使用可能な特定の文字（文字レパートリ）
- 記述文字とそれによって表現される言語
- レパートリ内のそれぞれの文字に割り当てられたコード値
- 文字エンティティを表現するために使用するコード体系

この章をとおして、これらの違いを説明します。

コード化する文字

キャラクタ・セットを決定する場合にまず選択することは、データベースへの格納をどの言語で行うかによります。キャラクタ・セットでコード化される文字は、これを表現する書込みシステムに依存します。

記述法

書込みシステムは、1 つの言語または 1 つの言語グループを表現するために使用されます。このマニュアルでは、書込みシステムは、表音的なものと表意的なものの 2 つに大きく分類されます。

表音的記述法

音声的書込みシステムは、1 つの言語に対応付けられたさまざまな音声を表現する記号で構成されます。ギリシア文字、ラテン文字、キリル文字およびデーヴァナーガリはすべて、アルファベットに基づいた表音的記述法の例です。アルファベットは、複数の言語を表現することができます。たとえば、ラテン・アルファベットでは、多くの西ヨーロッパ諸国の言語（フランス語、ドイツ語、英語など）を表現することができます。

表音的記述法と対応付けられた文字（アルファベット）は、文字レパートリが通常 256 文字より少ないので、一般的には 1 バイトでコード化されます。

表意的記述法

表意的記述法は、表音的記述法とは対照的に、言語の音声ではなく単語の意味を表す表意文字または象形文字で構成されます。中国語および日本語は、何万という表意文字に基づいた表意的記述法の例です。表意的記述法を使用する言語は、音節文字表を使用する場合もあります。音節文字表では、象形文字と一緒に音声情報を伝達するメカニズムが必要に応じて提供されています。たとえば、日本語には音節文字表が 2 つあり、通常、外来語や擬音語にはカタカナが使用されます。

表意的記述法と対応付けられた文字は、その文字レパートリが何万とあるので、一般的には複数バイトでコード化されます。

句読点、制御文字、数字および記号

言語スクリプトのコード化に加えて、句読点（カンマ、ピリオド、アポストロフィなど）、数字（アラビア数字の 0 ～ 9 など）、特殊記号（通貨記号、数学演算子など）、コンピュータの制御文字（キャリッジ・リターン、タブ、NULL など）のような特殊文字もコード化する必要があります。

書込み方向

欧文言語のほとんどは、ページの上から下へ向かって左から右へ書き込まれます。東アジア地域言語は、通常、ページの右から左に向かって上から下へ書き込まれます。欧文言語を翻訳した専門書では例外が多く見られます。

もう 1 つの考慮事項は、アラビア語およびヘブライ語では数字が逆に書き込まれることです。つまり、テキストが右から左に書かれている場合でも、その文中の数字は左から右に向かって書かれます。たとえば、「I wrote 32 books」は「skoob 32 etorw I」となります。書込み方向にかかわらず、データは論理順序で格納されます。論理順序とは、入力している言語が画面上でどのように表示されるかではなく、その言語が使用する順序を意味します。

キャラクタ・セットがサポートする言語数

さまざまなキャラクタ・セットは、さまざまな文字レパートリをサポートします。一般的に、キャラクタ・セットは特定の書込みスクリプトに基づいているので、さまざまな言語をサポートすることができます。キャラクタ・セットがはじめて米国で開発されたとき、それらは次の組込み文字レパートリに制限されていました。

- 大文字と小文字の英文字（A ～ Z および a ～ z）
- アラビア数字（0 ～ 9）
- 句読点文字（! " # $ % & ( ) * + , - . / : ; < = > ? @）
- 共通する一部の制御文字（NULL、キャリッジ・リターン、削除を含む）

たとえば、ASCII と IBM EBCDIC キャラクタ・セットでは、同じ文字レパートリがサポートされていますが、文字によっては異なるコード値が割り当てられている場合があります。表 3-2「7 ビット ASCII コード化キャラクタ・セット」に、ASCII がどのようにコード化されているかを示します。行および列の見出しは、16 進数字を示しています。文字のコード値を調べるには、列の数字の次に行の数字を読みます。たとえば、A の値は 0x41 です。

表 3-2 7 ビット ASCII コード化キャラクタ・セット

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	'	P
1	SOH	DCI	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	TAB	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

世界中のユーザーの高まる要求を満たすために、キャラクタ・セットは年々発展し、英語 1 か国語のみではなくそれ以上の言語をサポートするようになりました。他の言語をサポートする新規のキャラクタ・セットが素早く作成されました。一般的に、このような新規のキャラクタ・セットでは同じ文字に基づいた関連のある言語グループがサポートされています。たとえば、ISO 8859 キャラクタ・セット・シリーズは、多くの国家規格または地域標準に準拠して作成され、さまざまなヨーロッパ諸国言語をサポートしています。

表 3-3 ISO 8859 キャラクタ・セット

規格	サポートしている言語
ISO 8859-1	西ヨーロッパ諸国の言語（アラビア語、バスク語、ブルトン語、カタロニア語、デンマーク語、オランダ語、英語、フェロー語、フィンランド語、フランス語、ドイツ語、グリーンランド語、アイスランド語、アイルランドのゲール語、イタリア語、ラテン語、ルクセンブルク語、ノルウェー語、ポルトガル語、レートロマンス語、スコットランドのゲール語、スペイン語、スウェーデン語）
ISO 8859-2	西ヨーロッパ諸国の言語（アルバニア語、クロアチア語、チェコ語、英語、ドイツ語、ハンガリー語、ラテン語、ポーランド語、ルーマニア語、スロバキア語、スロベニア語、セルビア語）
ISO 8859-3	南東ヨーロッパ諸国の言語（アフリカーンス語、カタロニア語、オランダ語、英語、エスペラント語、ドイツ語、イタリア語、マルタ語、スペイン語、トルコ語）
ISO 8859-4	北ヨーロッパ諸国の言語（デンマーク語、英語、エストニア語、フィンランド語、ドイツ語、グリーンランド語、ラテン語、ラトビア語、リトアニア語、ノルウェー語、サーミ語、スロベニア語、スウェーデン語）
ISO 8859-5	東ヨーロッパ諸国の言語（キリルベース : ブルガリア語、ベロルシア語、マケドニア語、ロシア語、セルビア語、ウクライナ語）
ISO 8859-6	アラビア語
ISO 8859-7	ギリシア語
ISO 8859-8	ヘブライ語
ISO 8859-9	西ヨーロッパ諸国の言語（アラビア語、バスク語、ブルトン語、カタロニア語、コーンウォール語、デンマーク語、オランダ語、英語、フィンランド語、フランス語、フリジア語、ガリシア語、ドイツ語、グリーンランド語、アイルランドのゲール語、イタリア語、ラテン語、ルクセンブルク語、ノルウェー語、ポルトガル語、レートロマンス語、スコットランドのゲール語、スペイン語、スウェーデン語、トルコ語）
ISO 8859-10	北ヨーロッパ諸国の言語（デンマーク語、英語、エストニア語、フェロー語、フィンランド語、ドイツ語、グリーンランド語、アイスランド語、アイルランドのゲール語、ラテン語、リトアニア語、ノルウェー語、サーミ語、スロベニア語、スウェーデン語）
ISO 8859-15	西ヨーロッパ諸国の言語（アラビア語、バスク語、ブルトン語、カタロニア語、デンマーク語、オランダ語、英語、エストニア語、フェロー語、フィンランド語、フランス語、フリジア語、ガルシア語、ドイツ語、グリーンランド語、アイスランド語、アイルランドのゲール語、イタリア語、ラテン語、ルクセンブルク語、ノルウェー語、ポルトガル語、レートロマンス語、スコットランドのゲール語、スペイン語、スウェーデン語）

キャラクタ・セットは、多言語を制限付きでサポートしてきました。この制限とは、類似する文字に基づく言語グループに限りサポートしてきたという意味です。

最近では、制限のないまたはユニバーサルなキャラクタ・セットを使用して表現できる文字データについて、境界および制限をなくそうとする努力が行われています。Unicode はこのようなユニバーサル・キャラクタ・セットの 1 つで、現代および古代世界の主要文字のほとんどがこれに含まれています。Unicode キャラクタ・セットでは、約 39,000 文字の文字レパートリがサポートされており、これは増え続けています。

文字のコード化方法

さまざまなタイプのコード体系が、コンピュータ業界で作成されてきました。これらのコード体系にはさまざまなパフォーマンス特性があります。コード体系によっては、文字データを操作する場合のデータベース・スキーマおよびアプリケーションの開発要件に影響を及ぼす場合があります。そのため、選択したキャラクタ・セットが使用するコード体系の特徴を十分に理解しておく必要があります。一般的に、選択するキャラクタ・セットは次のいずれかのタイプのコード体系を使用します。

シングルバイト・コード体系

シングルバイト・コード体系は、最も効率的なコード体系です。シングルバイトの 1 文字は 1 バイトで表現されるので、文字を表現するのに必要な領域の合計量は最も小さく、これを処理したりプログラミングで使用したりすることは容易です。

7 ビット・コード体系

シングルバイト 7 ビット・コード体系では、128 文字までの文字を定義でき、通常、1 つの言語だけをサポートします。最も一般的なシングルバイト・キャラクタ・セットは、ASCII（American Standard Code for Information Interchange）と US EBCDIC で、コンピュータ処理の初期の頃から使用されていました。

8 ビット・コード体系

シングルバイト 8 ビット・コード体系では、256 文字までの文字を定義でき、通常、1 つの関連言語グループをサポートします。たとえば、ISO 8859/1 は、西ヨーロッパ諸国の言語を多数サポートしています。

	0	1	2	3	4	5	6	7	A	B	C	D	E	F
0	NUL	DLE	SP	0	@	P	`	p	NBSP	°	À	Ð	à	ð
1	SOH	DC1	!	1	A	Q	a	q	¡	±	Á	Ñ	á	ñ
2	STX	DC2	"	2	B	R	b	r	¢	²	Â	Ò	â	ò
3	ETX	DC3	#	3	C	S	c	s	£	³	Ã	Ó	ã	ó
4	EOT	DC4	$	4	D	T	d	t	¤	´	Ä	Ô	ä	ô
5	ENQ	NAK	%	5	E	U	e	u	¥	µ	Å	Õ	å	õ
6	ACK	SYN	&	6	F	V	f	v	¦	¶	Æ	Ö	æ	ö
7	BEL	ETB	'	7	G	W	g	w	§	·	Ç	×	ç	÷
8	BS	CAN	(	8	H	X	h	x	¨	¸	È	Ø	è	ø
9	HT	EM	)	9	I	Y	i	y	©	¹	É	Ù	é	ù
A	NL	SUB	*	:	J	Z	j	z	ª	º	Ê	Ú	ê	ú
B	VT	ESC	+	;	K	[	k	{	«	»	Ë	Û	ë	û
C	NP	FS	,	<	L	\	l	|	¬	¼	Ì	Ü	ì	ü
D	CR	GS	-	=	M	]	m	}		½	Í	Ý	í	ý
E	SO	RS	.	>	N	^	n	~	®	¾	Î	Þ	î	þ
F	SI	US	/	?	O	_	o	DEL	¯	¿	Ï	ß	ï	ÿ

マルチバイト・コード体系

マルチバイト・コード体系では、中国語または日本語のようなアジア諸国の言語で使用されている表意文字をサポートする必要があります。これは、中国語や日本語などが何千という文字を使用するためです。このようなコード体系は、固定のバイト数または可変のバイト数を使用して 1 文字を表現します。

固定幅コード体系

固定幅マルチバイト・コード体系では、それぞれの文字が n バイト固定で表現されます。この場合、n>=2 です。

可変幅コード体系

可変幅コード体系は、1 バイト以上を使用して 1 つの文字を表現します。マルチバイト・コード体系には、特定のビットを使用して、1 文字を表現するために使用するバイト数を示すものもあります。たとえば、1 文字を表現するために使用する最大のバイト数が 2 バイト

であった場合、あるバイトがシングルバイト文字の一部であるか、またはダブルバイト文字の 2 番目のバイトであるかを示すために、この暗示ビットがトグルされます。他の体系では、制御コードによって、シングルバイトがダブルバイト文字と区別されます。シフトアウト・コードは、シフトイン・コードが検出されるまでは、後続のバイトがダブルバイト文字であることを示します。

Oracle のキャラクタ・セットの命名規則

キャラクタ・セット名には次の命名規則が使用されます。

```
<language_or_region><#_of_bits_representing_a_char><standard_name>[S] [C] [FIXED]
```

次に例を示します。

US7ASCII	US 7 ビット ASCII キャラクタ・セット
WE8ISO8859P1	西欧 8 ビット ISO 8859 Part 1 キャラクタ・セット
JA16SJIS	日本語 16 ビット・シフト JIS キャラクタ・セット

キャラクタ・セット名の最後にオプションの「S」または「C」を付けることによって、サーバー（S）またはクライアント（C）でのみ使用できるキャラクタ・セットであることを区別する場合もあります。

Macintosh のプラットフォームでは、サーバー・キャラクタ・セットが常に使用されます。Macintosh のクライアント・キャラクタ・セットは、現在使用されていません。EBCDIC プラットフォームでは、使用可能であれば、「S」バージョンがサーバーで、「C」バージョンがクライアントで使用されます。

キャラクタ・セット名の最後にオプションの「FIXED」を付けることによって、それが固定幅マルチバイト・コードであることを示します。

Oracle データベース・キャラクタ・セットの選択のヒント

Oracle では、次の項目でデータベース・キャラクタ・セットを使用します。

- CHAR、VARCHAR2、CLOB および LONG 列に格納されるデータ
- 表名、列名、PL/SQL 変数などの識別子
- SQL と PL/SQL プログラム・ソースの入力、およびその格納

データベース用の Oracle キャラクタ・セットを選択する場合は、次の 4 つの項目について考慮する必要があります。

1. データベースがサポートしなければならない言語
2. システム・リソースとアプリケーションの相互運用
3. パフォーマンスとの関係
4. 制限事項

いくつかのキャラクタ・セットが現在の言語要件を満たしているかもしれませんが、将来の言語要件も考慮してください。将来、さまざまな言語をサポートする必要があると分かっている場合は、サポート範囲の広いキャラクタ・セットを採用しておくと、後でキャラクタ・セットを移行する必要がなくなります。付録 A「ロケール・データ」に示されている Oracle キャラクタ・セットは、そのキャラクタ・セットがサポートする言語および地域に従って名前が付けられています。地域が不明なキャラクタ・セット（たとえば ISO キャラクタ・セット）の場合は、言語によって明示的に示されています。コード化された実際の文字を参照することもできます。実際のコード・ページは、このマニュアルでは説明しませんが、ほとんどのコード・ページは、各国の、国際的なまたはベンダー製品ドキュメントに基づいているか、あるいは標準ドキュメントで参照できます。

システム・リソースとアプリケーションの相互運用

データベースが実際の文字データをメンテナンスし処理する際、オペレーティング・システムの他に依存しなければならないリソースがあります。たとえば、オペレーティング・システムによって、選択したキャラクタ・セットに対応するフォントが提供されます。また、希望する言語およびアプリケーション・ソフトウェアをサポートする入力メソッドは、特定のキャラクタ・セットと互換性がなければなりません。

キャラクタ・セットがオペレーティング・システム上で使用可能で、ご使用のアプリケーションによって処理されるという、シームレスな統合を確立することが理想です。

キャラクタ・セット変換

オペレーティング・システムで使用可能なキャラクタ・セットと異なるキャラクタ・セットを選択した場合、データベース・キャラクタ・セットをオペレーティング・システム・キャラクタ・セットに変換することができます。ただし、このキャラクタ・セット変換では多少のオーバーヘッドが生じるので、オペレーティング・システム・キャラクタ・セットに同等の文字レパートリが含まれていることを確認して、データが損失しないようにしてください。

また、キャラクタ・セット変換でデータが損失する場合もあります。たとえば、キャラクタ・セット A をキャラクタ・セット B に変換する場合、変換先のキャラクタ・セット（B）には、A と同じキャラクタ・セット・レパートリが含まれていなければなりません。キャラクタ・セット B で使用できない文字があると、それらは置換文字に変換されます。置換文字には、多くの場合、「?」または言語的に関連する文字が使用されます。たとえば、ä（ウムラウト付きの a）は「a」に変換されます。分散環境の場合には、類似した文字レパートリを持つキャラクタ・セットを使用してデータが損失しないようにしてください。

キャラクタ・セット変換では、データがクライアントに到達するまでに文字列が何度もバッファ間でコピーされる場合があります。したがって、クライアントとサーバーで同じキャラクタ・セットを使用すると、キャラクタ・セット変換を回避することができるので、最適なパフォーマンスを得ることができます。

データベース・スキーマ

文字データ型 CHAR および VARCHAR2 は、文字単位ではなくバイト単位で指定します。したがって、表定義で CHAR(20) と指定すると、文字データを格納するために 20 バイトが割り当てられます。

データベース・キャラクタ・セットがシングルバイト・コード体系を使用している場合、文字数がバイト数と同じであるので、このような指定は適切に処理されます。データベース・キャラクタ・セットがマルチバイト文字コード体系を使用している場合は、このような一致は生じません。つまり、1 文字が 1 バイト以上で構成されるので、バイト数が文字数と等しくなくなってしまっています。したがって、特定の文字数について予測できる最大バイト数を見込んで、列幅を慎重に選択する必要があります。

パフォーマンスとの関係

選択したキャラクタ・セットに依存するさまざまなコード体系を処理する場合、生じるパフォーマンスのオーバーヘッドもさまざまです。最適なパフォーマンスを得るために、キャラクタ・セット変換を回避し、希望する言語にとって最も効果的なコード化を使用するキャラクタ・セットを選択するようにしてください。シングルバイト・キャラクタ・セットは、パフォーマンスの点ではマルチバイト・キャラクタ・セットよりも適しています。また、領域要件という点においてもシングルバイト・キャラクタ・セットが最も効率的です。

制限事項

現在、固定幅マルチバイト・キャラクタ・セットは Oracle データベース・キャラクタ・セットには選択できません。特に、次のキャラクタ・セットはデータベース・キャラクタ・セットとして使用できません。

JA16EUCFIXED
ZHS16GBKFIXED
JA16DBCSFIXED
KO16DBCSFIXED
ZHS16DBCSFIXED
JA16SJISFIXED
ZHT32TRISFIXED

Oracle NCHAR キャラクタ・セットの選択のヒント

場合によっては、データベース用に代替キャラクタ・セットを選択することができます。これは、大量の文字処理操作を行う場合に、またはプログラミングを容易にするために別の文字コード体系のレパートリがより望ましいことがあるためです。特に、次のデータ型が代替キャラクタ・セットとして使用できます。

- NCHAR
- NVARCHAR2
- NCLOB

NCHAR キャラクタ・セットを指定すると、NCHAR、NVARCHAR2 および NCLOB 列で使用するためのデータベース・キャラクタ・セットの代替キャラクタ・セットを指定できます。NCHAR は固定幅のマルチバイト・コード体系をサポートできるので、NCHAR キャラクタ・セットは可変幅マルチバイト・データベース・キャラクタ・セットを使用するユーザーには特に便利です。これに対して、データベース・キャラクタ・セットは固定幅のマルチバイト・コード体系をサポートしていません。固定幅マルチバイト・コードを使用すると、可変幅のマルチバイト・コードを使用する場合に比べて次の利点があります。

- NCHAR、NVARCHAR2 および NCLOB 列での文字列の処理パフォーマンスが最適化される。
- 可変幅マルチバイト・キャラクタ・セットに対し、固定幅マルチバイト・キャラクタ・セットを使用するとプログラミングが容易である。

NCHAR キャラクタ・セットを選択するときは、NCHAR の文字レパートリがデータベースのキャラクタ・セット・レパートリと同等であるか、またはそのサブセットであることを確認する必要があります。

注意 : すべての SQL コマンドでは、NCHAR キャラクタ・セットではなくデータベース・キャラクタ・セットが使用されます。したがって、リテラルはデータベース・キャラクタ・セットでのみ指定できます。

データベース・スキーマ

データ型 NCHAR、NVARCHAR2 および NCLOB の幅は、使用するコード体系に依存するバイト数または文字数という言葉で置き換えることができます。NCHAR キャラクタ・セットが可変幅マルチバイト・コード体系を使用している場合、その幅指定はバイト数になります。NCHAR キャラクタ・セットが固定幅マルチバイト・コード体系を使用している場合、その幅指定は文字数になります。たとえば、可変幅マルチバイト・キャラクタ・セットの JA16EUC を使用している場合、NCHAR(20) では 20 バイトが割り当てられます。これに対して、固定幅マルチバイト・キャラクタ・セットの JA16EUCFIXED を使用している場合、NCHAR(20) では 40 バイトが割り当てられます。

パフォーマンスとの関係

一部の文字列操作は、各国文字キャラクタ・セットとして固定幅キャラクタ・セットを選択すると高速になります。たとえば、SQL LIKE 演算子などの文字列集中操作を NCHAR の固定幅列で使用すると、マルチバイト列の LIKE 演算子よりも優れたパフォーマンスが得られます。可能な使用例は、次のとおりです。

データベース・キャラクタ・セット	NCHAR キャラクタ・セット
JA16EUC	JA16EUCFIXED

制限事項

SQL テキストは、NCHAR キャラクタ・セットではなく、データベース・キャラクタ・セットによってのみ表現されるので、データベース・キャラクタ・セットと同等またはそのサブセットにあたる文字レパートリを持つ NCHAR キャラクタ・セットを選択する必要があります。

異なるコード体系に関する考慮事項

コード体系を取り扱う場合の考慮事項がいくつかあります。

固定幅および可変幅のキャラクタ・セットが混在する場合の注意

固定幅マルチバイト・キャラクタ・セットは文字単位で測定されますが、可変幅キャラクタ・セットはバイト単位で測定されます。このため、1 つのプラットフォーム上では各国キャラクタ・セットとして固定幅マルチバイト・キャラクタ・セットを使用し、なおかつ別のプラットフォーム上では可変幅キャラクタ・セットを使用する場合には注意が必要です。

たとえば、%TYPE または指定したタイプを使用して 1 つのプラットフォーム上のある項目を宣言する際に、他方のプラットフォームから取得した項目の宣言情報を使用したとします。この場合には、受け取る制約の枠が小さすぎるために、データをサポートできない可能性があります。一例として、固定幅マルチバイト・セットを使用しているプラットフォーム上では、NCHAR(10) によって 10 文字分の領域が割り当てられます。しかし、%TYPE または指定されたタイプを使用して、これと同様に入力された項目を他方のプラットフォーム上に作成した場合、割り当てられるのは 10 バイトのみです。通常、これでは 10 文字には不十分です。この問題を回避するためには、次のいずれかに従ってください。

- 異なるプラットフォーム上での各国キャラクタ・セットとして、固定幅と可変幅のマルチバイト・キャラクタ・セットを混在させない。
- 異なるプラットフォーム上での各国キャラクタ・セットとして固定幅と可変幅のマルチバイト・キャラクタ・セットを混在させる場合には、制約値が比較的大きい可変長タイプ宣言を使用する。

マルチバイト・キャラクタ・セットのデータの格納

文字データ型 CHAR および VARCHAR2 の幅は、文字ではなくバイトで指定します。したがって、表定義で CHAR(20) と指定すると、文字データを格納するために 20 バイトが割り当てられます。

データベースのキャラクタ・セットがシングルバイトの場合は、文字数とバイト数は同じになります。データベースのキャラクタ・セットがマルチバイトの場合、一般に、このような一致は生じません。1 つの文字が 1 バイトで構成されることも、複数バイトで構成されることもあります。これは、特定のマルチバイト・コード体系とシフトイン / シフトアウト制御コードの有無によって異なります。したがって、特定の文字数について予測できる最大バイト数を見込んで、列幅を慎重に選択する必要があります。

文字データ型 NCHAR および NVARCHAR2 の幅は、各国キャラクタ・セットが固定幅マルチバイトである場合には、文字で指定します。固定幅マルチバイトでない場合には、バイトで指定します。

幅の小さいキャラクタ・セットを使用した場合に発生するもう 1 つのパフォーマンス上の問題はスペース効率（スピード）です。キャラクタ・セットの選択時（可変幅または固定幅）には、この 2 つの点をどのように考慮するのかを判断しなければなりません。

データベース・オブジェクトの命名

Oracle を使用すると、データベース・オブジェクトに名前を付けることができます。

名前およびテキストを表すために使用するキャラクタ・セットに関する制限事項

表 3-4 に、Oracle で名前その他のテキストを表すために使用できるキャラクタ・セットに関する制限事項を示します。

表 3-4 名前およびテキストを表すために使用するキャラクタ・セットに関する制限事項

名前	シングルバイト固定	可変幅	マルチバイト固定幅キャラクタ・セット	コメント
コメント	可能	可能	可能	
データベース・リンク名	可能	不可	不可	
データベース名	可能	不可	不可	
ファイル名（データ・ファイル、ログファイル、制御ファイル、初期化パラメータ・ファイル）	可能	不可	不可	
インスタンス名	可能	不可	不可	
ディレクトリ名	可能	不可	不可	
キーワード	可能	不可	不可	英語でのみ表現可能
Recovery Manager ファイル名	可能	不可	不可	
ロールバック・セグメント名	可能	不可	不可	ROLLBACK_SEGMENTS パラメータでは NLS は未サポート
ストアド・スクリプト名	可能	可能	不可	
表領域名	可能	可能	不可	

LOB データ（LOB、BLOB、CLOB および NCLOB）などのサポートされている文字列形式およびキャラクタ・セットについては、表 3-6 を参照してください。

データベースで使用される文字コード体系は、データベースの作成時に CREATE DATABASE 文の一部として定義されます。データ・ディクショナリ内の列を含めて、CHAR、CLOB、VARCHAR2 および LONG 型のデータ列はすべて、データベースのキャラクタ・セットの文字を使用して格納されます。さらに、選択するデータベース・キャラクタ・セットによって、どの文字でデータベース内のオブジェクトを指定できるかが決定します。NCHAR、NCLOB および NVARCHAR2 型のデータ列では、各国キャラクタ・セットを使用します。

一度データベースを作成すると、再作成しない限りキャラクタ・セットの選択は変更できません。したがって、どのキャラクタ・セットを使用するかを慎重に考慮することが大切です。データベースのキャラクタ・セットは、常に、クライアントのオペレーティング・システムのネイティブ・キャラクタ・セットと同等か、そのスーパーセットでなければなりません。データベースにアクセスするクライアント・アプリケーションが使用するキャラクタ・セットによって、通常データベース用として最も適したスーパーセットが決定されます。

すべてのクライアント・アプリケーションが同じキャラクタ・セットを使用する場合、そのキャラクタ・セットが、通常、データベースのキャラクタ・セットとして選択されます。複数のクライアント・アプリケーションが異なるキャラクタ・セットを使用するときは、データベースのキャラクタ・セットが、クライアントのすべてのキャラクタ・セットと同等か、そのスーパーセットでなければなりません。これによって、クライアントのキャラクタ・セットからデータベースのキャラクタ・セットへの変換時に、すべての文字が確実に表示されます。

異なるキャラクタ・セットを使用する端末でクライアント・アプリケーションが動作するとき、クライアント・アプリケーションの文字を、データベースのキャラクタ・セットに変換（またはその逆）する必要があります。この変換は自動的に実行されますが、クライアント・アプリケーション側で意識する必要はありません。クライアント・アプリケーションが使用するキャラクタ・セットは、NLS_LANG パラメータで定義します。同様に、各国キャラクタ・セット・データとして使用するキャラクタ・セットは NLS_NCHAR で定義します。

データ型の概要とサポートしているコード体系

表 3-5 に、さまざまなデータ型と対応付けて、サポートしているコード体系を示します。

表 3-5 サポートしているデータ型のコード体系

データ型	シングルバイト	マルチバイト可変幅	マルチバイト固定幅
CHAR	可能	可能	不可
NCHAR	可能	可能	可能
BLOB	可能	可能	可能
CLOB	可能	可能	不可
NCLOB	可能	可能	可能

表 3-6 に、抽出データ型（ADT）と対応付けて、サポートしているデータ型を示します。

表 3-6 サポートしている抽出データ型のデータ型

抽出データ型	CHAR	NCHAR	BLOB	CLOB	NCLOB
オブジェクト	可能	不可	可能	可能	不可
コレクション	可能	不可	可能	可能	不可

注意 : BLOB では、文字は一連のバイト順序として処理されます。データは、NLS 依存操作の対象にはなりません。

データベース作成後のキャラクタ・セットの変更

既存のデータベース・キャラクタ・セットを変更することが必要な場合があります。たとえば、ご使用のデータベースでサポートしなければならない言語数が増加する場合があります。ほとんどの場合、全体のエクスポート / インポートを行って、データをすべて新しいキャラクタ・セットに正しく変換する必要があります。ただし、新しいキャラクタ・セットが現行のキャラクタ・セットの完全なスーパーセットである場合に限り、ALTER DATABASE CHARACTER SET を使用してデータベース・キャラクタ・セットを効率よく変更することができます。

ソース・キャラクタ・セットのコード・ポイントがすべてターゲット・キャラクタ・セットで使用可能で、そのコード・ポイントの値が一致する場合に限り、ターゲットのキャラクタ・セットは完全なスーパーセットと言えます。たとえば、US7ASCII は WE8ISO8859P1、AL24UTFFSS および UTF8 の完全なスーパーセットであるため、次のような移行シナリオでは、ALTER DATABASE CHARACTER SET コマンドを利用できます。

現行キャラクタ・セット	新規キャラクタ・セット	新規キャラクタ・セットは 完全なスーパーセットか？
US7ASCII	WE8ISO8859P1	はい
US7ASCII	ALT24UTFFSS	はい
US7ASCII	UTF8	はい

警告 : データベース・キャラクタ・セットを完全なスーパーセットでないキャラクタ・セットに変更しようとすると、データが損失および破壊する可能性があります。完全なスーパーセットでないキャラクタ・セットに新たに移行する場合は必ず、データの整合性を保証するためにエクスポート / インポートを実行してください。ALTER DATABASE [NATIONAL] CHARACTER SET 文はロール・バックされないので、必ずデータベース全体をバックアップしてからこのコマンドを使用してください。構文は、次のとおりです。

```
ALTER DATABASE [<db_name>] CHARACTER SET <new_character_set>;
ALTER DATABASE [<db_name>] NATIONAL CHARACTER SET <new_NCHAR_character_set>;
```

データベース名はオプションです。キャラクタ・セット名は、引用符を付けずに指定しなければなりません。たとえば、次のように指定します。

```
ALTER DATABASE CHARACTER SET WE8ISO8859P1;
```

データベース・キャラクタ・セットを変更するには、次の手順を実行します。すべての手順が絶対に必要であるというわけではありませんが、すべての実行をお薦めします。

```
SQL> SHUTDOWN IMMEDIATE;   -- or NORMAL
    <do a full backup>

SQL> STARTUP MOUNT;
SQL> ALTER SYSTEM ENABLE RESTRICED SESSION;
SQL> ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0;
SQL> ALTER DATABASE OPEN;
SQL> ALTER DATABASE CHARACTER SET <new_character_set_name>;
SQL> SHUTDOWN IMMEDIATE;   -- or NORMAL
SQL> STARTUP;
```

各国キャラクタ・セットを変更する場合は、ALTER DATABASE CHARACTER SET 文を ALTER DATABASE NATIONAL CHARACTER SET 文に置き換えます。必要であれば、両方のコマンドを一緒に発行することができます。

キャラクタ・セットのカスタマイズ

キャラクタ・セットを調整して、特定のユーザーのニーズに合わせることが必要な場合があります。Oracle8*i* では、ユーザーは既存のコード化キャラクタ・セットの定義を拡張して、独自のニーズに合わせることができます。ユーザー定義文字（UDC）は、多くの場合、特殊文字の表現をコード化するために使用されます。

- 正式名
- 既存のキャラクタ・セット規格で定義されていない旧漢字
- ベンダー固有の文字
- ユーザーが新たに定義した記号、文字など

この項では、Oracle がどのように UDC をサポートしているかについて説明します。説明する項目は次のとおりです。

- ユーザー定義文字とキャラクタ・セット
- Oracle のキャラクタ・セット変換アーキテクチャの理解
- Unicode 2.0 の Private Use Area
- UDC クロス・リファレンス

ユーザー定義文字とキャラクタ・セット

一般的に、ユーザー定義文字は東アジア諸国のキャラクタ・セットでサポートされています。このような東アジア諸国のキャラクタ・セットには、ユーザー定義文字を使用するために、コード・ポイントの予約域が少なくとも 1 つ用意されています。たとえば、日本のシフト JIS では、次の 1880 のコード・ポイントが UDC 用に保持されています。

日本語シフト JIS の UDC 範囲	コード・ポイント数
0xf040 ～ 0xf07e、0xf080 ～ 0xf0fc	188
0xf140 ～ 0xf17e、0xf180 ～ 0xf1fc	188
0xf240 ～ 0xf27e、0xf280 ～ 0xf2fc	188
0xf340 ～ 0xf37e、0xf380 ～ 0xf3fc	188
0xf440 ～ 0xf47e、0xf480 ～ 0xf4fc	188
0xf540 ～ 0xf57e、0xf580 ～ 0xf5fc	188
0xf640 ～ 0xf67e、0xf680 ～ 0xf6fc	188
0xf740 ～ 0xf77e、0xf780 ～ 0xf7fc	188
0xf840 ～ 0xf87e、0xf880 ～ 0xf8fc	188
0xf940 ～ 0xf97e、0xf980 ～ 0xf9fc	188

次に示す Oracle キャラクタ・セットには、ユーザー定義文字のサポート用としてあらかじめ定義されている範囲があります。

表 3-7 UDC と Oracle キャラクタ・セット

キャラクタ・セット名	使用可能な UDC コード・ポイント数
JA16DBCS	4370
JA16DBCSFIXED	4370
JA16EBCDIC930	4370
JA16SJIS	1880
JA16SJISFIXED	1880
JA16SJISYEN	1880
KO16DBCS	1880
KO16DBCSFIXED	1880
KO16MSWIN949	1880
ZHS16DBCS	1880
ZHS16DBCSFIXED	1880
ZHS16GBK	2149
ZHS16GBKFIXED	2149

表 3-7 UDC と Oracle キャラクタ・セット

キャラクタ・セット名	使用可能な UDC コード・ポイント数
ZHT16DBCS	6204
ZHT16MSWIN950	6217

Oracle のキャラクタ・セット変換アーキテクチャ

特定の文字を表すコード・ポイントの値は、キャラクタ・セットによって異なります。たとえば、日本語の漢字である

亜

は、さまざまな日本語キャラクタ・セットによって次のようにコード化されます。

キャラクタ・セット	Unicode	JA16SJIS	JA16EUC	JA16DBCS
亜 の文字の値	0x4E9C	0x88F9	0xB0A1	0x4867

Oracle では、すべてのキャラクタ・セットが Unicode 2.0 のコード・ポイントによって定義されています。つまり、それぞれの文字は Unicode 2.0 のコード値で定義されています。文字変換は、中間フォームとして Unicode を使用してユーザーが意識することなく行われます。たとえば、JA16SJIS であるクライアントが JA16EUC であるデータベースに接続した場合、文字の

亜

は、JA16SJIS のクライアントで入力された値 0x88F9 が内部的に Unicode の値 0x4E9C に変換され、JA16EUC の値 0xB0A1 にさらに変換されます。

Unicode 2.0 の Private Use Area

Unicode 2.0 では、Private Use Area（PUA）用に 0xE000 ～ 0xF8FF の範囲が予約されています。PUA は、エンド・ユーザーまたはベンダーがプライベートで使用する文字を定義するためのものです。

UDC は、標準文字と同様に中間フォームとして Unicode 2.0 の PUA を使用して、2 つの Oracle キャラクタ・セット間で変換されます。

UDC クロス・リファレンス

日本語キャラクタ・セット、韓国語キャラクタ・セット、簡易字中国語キャラクタ・セットおよび繁体字中国語キャラクタ・セットでの UDC クロス・リファレンスは、次の配布セットに含まれています。

```
${ORACLE_HOME}/ocommon/nls/demo/udc_ja.txt
${ORACLE_HOME}/ocommon/nls/demo/udc_ko.txt
${ORACLE_HOME}/ocommon/nls/demo/udc_zhs.txt
${ORACLE_HOME}/ocommon/nls/demo/udc_zht.txt
```

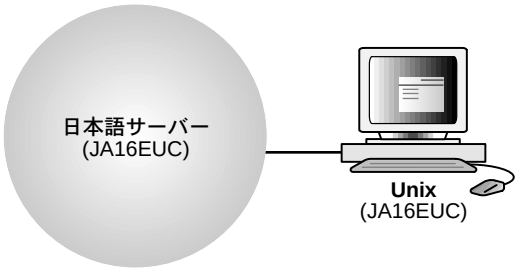
ユーザー定義文字をいくつかのオペレーティング・システムに登録する場合、これらのクロス・リファレンスが役に立ちます。たとえば、新規 UDC を日本語シフト JIS のオペレーティング・システムと日本語 IBM ホスト・オペレーティング・システムの両方に登録する場合、この新規 UDC に対して、シフト JIS のオペレーティング・システムでは 0xF040、IBM ホスト・オペレーティング・システムでは 0x6941 を採用することができます。これによって、JA16SJIS および JA16DBCS 間での変換が正しく行われるようになります。シフト JIS の UDC 値 0xF040 と IBM ホストの UDC 値 0x6941 は、UDC クロス・リファレンスでは同じ Unicode PUA 値 0xE000 にマップされています。

キャラクタ・セット定義ファイルのカスタマイズ方法の詳細は、付録 B「ロケール・データのカスタマイズ」を参照してください。

1 か国語データベースの例

クライアントとサーバーで同一のキャラクタ・セット

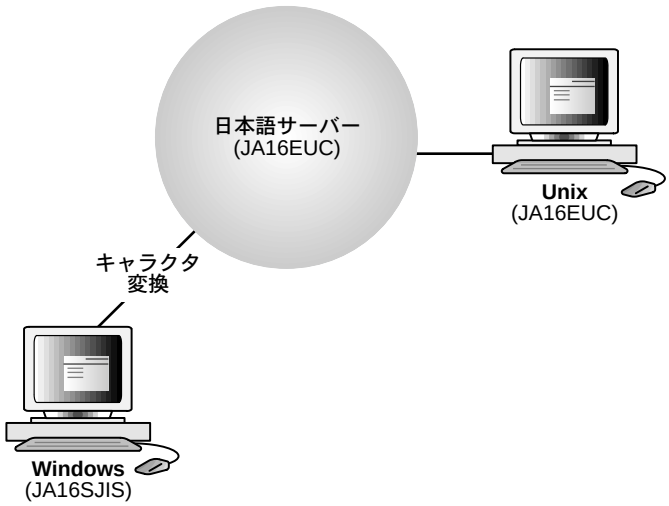
次に、最も簡単な NLS データベースの設定の例を示します。クライアントとサーバーの言語環境が同じで、両方で同じ文字コードを使用しているとします。1 か国語だけの使用例では、キャラクタ・セット変換に関連するオーバーヘッドが回避されるので、高速な応答が得られるという利点があります。



キャラクタ・セット変換

キャラクタ・セット変換は、多くの場合、クライアント / サーバー・コンピューティング環境で必須です。これは、クライアント・アプリケーションがサーバーと異なるコンピュータ・プラットフォーム上に常駐し、サーバーとクライアントの両方のプラットフォームで同じ文字コード体系が使用されないような環境です。クライアントとサーバー間で渡される文字データは、2 つのコード体系間で変換される必要があります。文字の変換は、ユーザーが意識することなく自動的に実行されます。

すべてのキャラクタ・セット間で変換が可能です。たとえば、次の場合も可能です。



ただし、ターゲットのキャラクタ・セットにソースのキャラクタ・セットの文字データがすべて含まれていない場合は、置換文字が使用されます。たとえば、US7ASCII を使用しているサーバーとドイツ語の WE8ISO8859P1 であるクライアントでは、ドイツ語の文字 ß は ? に置換され、文字 ä は a に置換されます。

置換文字は、キャラクタ・セットの定義の一部として、特定の文字に対して定義できます。特定の置換文字を定義しないと、デフォルトの置換文字が使用されます。クライアントのキャラクタ・セットからデータベースのキャラクタ・セットに変換するときに置換文字を使用しないようにするには、サーバーのキャラクタ・セットは、クライアントのすべてのキャラクタ・セットと同等か、またはそのスーパーセットでなければなりません。前述の例では、サーバーのキャラクタ・セットは賢明な選択ではありません。ドイツ語データがサーバーに格納されることを想定すると、ドイツ語の文字をサポートするキャラクタ・セット（たとえば、WE8ISO8859P1）がサーバーとクライアントの両方で必要になります。

可変幅マルチバイトでは、キャラクタ・セット変換のオーバーヘッドが著しく生じる場合があります。ユーザーは、状況を慎重に評価した上でキャラクタ・セットを選択し、できる限りキャラクタ・セット変換が行われないようにする必要があります。データベースとクライアントに適切なキャラクタ・セットを採用することによって、キャラクタ・セット変換だけではなく、データの損失を防ぐことができます。

多言語データベースの例

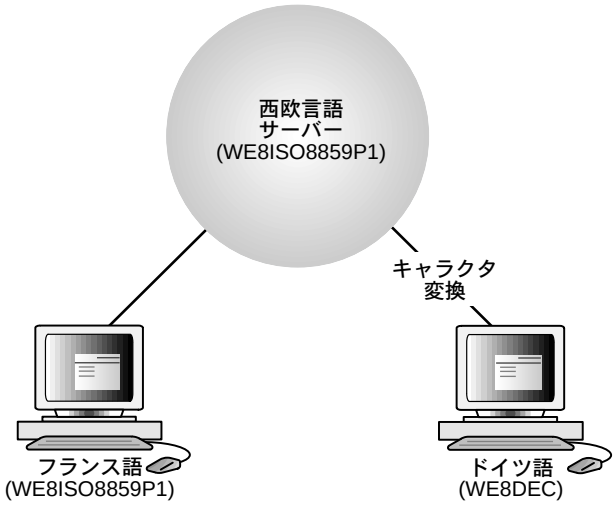
一部のキャラクタ・セットでは、複数の言語をサポートしています。たとえば、WE8ISO8859P1 は、次の西ヨーロッパ諸国の言語をサポートしています。

デンマーク語	フィンランド語	イタリア語	スウェーデン語
オランダ語	フランス語	ノルウェー語	
英語	ドイツ語	ポルトガル語	
フェロー語	アイスランド語	スペイン語	

これは、これらの言語がすべて、類似する書込みスクリプトに基づいているためです。このような状況は、制限付き多言語サポートと呼ばれます。制限付きと呼ばれるのは、このキャラクタ・セットが、関連する言語のグループに限りサポートしているためです。この場合、ISO8859-1 ではラテン語ベースの言語がサポートされています。

制限付き多言語サポート

次の図では、両方のクライアントがサーバーのデータにアクセスしています。

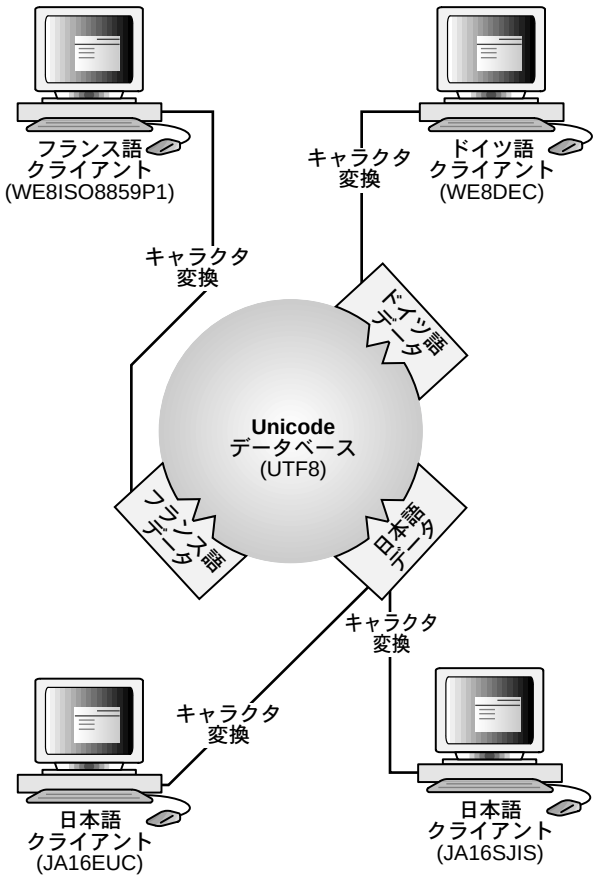


制限なし多言語サポート

しばしば、無制限の多言語サポートが要求され、Unicode などのユニバーサル・キャラクタ・セットがサーバーのデータベース・キャラクタ・セットとして必要になります。Unicode には、主なコード体系に UCS-2 と UTF-8 があります。UCS-2 は 2 バイト固定幅形式ですが、UTF-8 はマルチバイト可変幅形式です。Oracle8*i* では、UTF-8 形式がサポートされています。この拡張形式は、マルチバイト・キャラクタ・セットをすでにサポートしているクライアントでは意識する必要はありません。

UTF8 のデータベースとシングルバイト・キャラクタ・セット間でのキャラクタ・セット変換では、ごくわずかなオーバーヘッドが生じます。UTF8 とマルチバイト・キャラクタ・セット間での変換では、多少のオーバーヘッドが生じますが、変換によるデータ損失の問題はありません。

次の図に、データベースがどのように多種多様な言語をサポートするかについて示します。この場合、日本語、フランス語およびドイツ語のすべてのクライアントが、Unicode キャラクタ・セットに準拠している同一のデータベースにアクセスしています。


フランス語
クライアント
(WE8ISO8859P1)
キャラクタ
変換
キャラクタ
変換
ドイツ語
クライアント
(WE8DEC)
ドイツ語
データ
Unicode
データベース
(UTF8)
フランス語
データ
日本語
データ
キャラクタ
変換
キャラクタ
変換
日本語
クライアント
(JA16EUC)
日本語
クライアント
(JA16SJIS)

4

SQL プログラミング

この章では、NLS 環境での SQL プログラミングで役立つ次の情報について説明します。

ロケール依存の SQL 関数

その動作が NLS の規則に依存するすべての SQL 関数で、NLS パラメータを指定できます。それらの関数は、次のとおりです。

- TO_CHAR
- TO_DATE
- TO_NUMBER
- NLS_UPPER
- NLS_LOWER
- NLS_INITCAP
- NLSSORT

これらの関数にオプションの NLS パラメータを明示的に指定すると、関数の評価が、セッションで施行されている NLS パラメータに依存しないようにできます。この機能は、数字と日付を文字列リテラルとして含む SQL 文では重要となります。

たとえば、次の問合せは、日付に対する言語が米語に指定されている場合のみ正しく評価されます。

```
SELECT ENAME FROM EMP
WHERE HIREDATE > '1-JAN-91'
```

このような問合せが現行の日付の言語に依存しないようするには、次の文を使用します。

```
SELECT ENAME FROM EMP
WHERE HIREDATE > TO_DATE('1-JAN-91','DD-MON-YY',
   'NLS_DATE_LANGUAGE = AMERICAN')
```

このように、言語に依存しない SQL 文を、必要に応じて定義できます。たとえば、文字列リテラルが、ビュー、チェック制約、またはトリガー内の SQL 文にある場合には、このような文の指定が必要となることがあります。

すべての文字関数で、シングルバイトとマルチバイトの両方の文字をサポートします。単位を明示的に示した場合を除いて、文字関数は、バイト単位ではなく文字単位で動作します。

デフォルトの指定

ビューおよびトリガーを評価するとき、NLS 関数パラメータのデフォルト値は、セッションで現在有効な値から取得されます。チェック制約を評価するときは、デフォルト値は、データベースの作成時に指定された NLS パラメータによって設定されます。

パラメータの指定

NLS パラメータを SQL 関数に指定する構文は、次のとおりです。

```
'parameter = value'
```

次の NLS パラメータを指定できます。

- NLS_DATE_LANGUAGE
- NLS_NUMERIC_CHARACTERS
- NLS_CURRENCY
- NLS_ISO_CURRENCY
- NLS_SORT

次のように、特定の NLS パラメータのみが特定の SQL 関数に有効です。

SQL 関数	NLS パラメータ
TO_DATE	NLS_DATE_LANGUAGE NLS_CALENDAR
TO_NUMBER	NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY
TO_CHAR	NLS_DATE_LANGUAGE NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY NLS_CALENDAR
NLS_UPPER	NLS_SORT
NLS_LOWER	NLS_SORT
NLS_INITCAP	NLS_SORT
NLSSORT	NLS_SORT

次に、NLS パラメータの使用例を示します。

```
TO_DATE ('1-JAN-89', 'DD-MON-YY',
   'nls_date_language = American')

TO_CHAR (hiredate, 'DD/MON/YYYY',
   'nls_date_language = French')

TO_NUMBER ('13.000,00', '99G999D99',
   'nls_numeric_characters = ''.,''')

TO_CHAR (sal, '9G999D99L', 'nls_numeric_characters = ''.,''
   nls_currency = ''Dfl ''')

TO_CHAR (sal, '9G999D99C', 'nls_numeric_characters = '',.''
   nls_iso_currency = Japan')

NLS_UPPER (ename, 'nls_sort = Austrian')

NLSSORT (ename, 'nls_sort = German')
```

注意 : 言語によっては、さまざまな小文字が一連の大文字に対応したり、その逆の場合もあります。その結果、NLS_UPPER、NLS_LOWER および NLS_INITCAP の出力は、入力の長さと異なることがあります。

受け入れられないパラメータ

NLS_LANGUAGE および NLS_TERRITORY は、NLSSORT 以外の SQL 関数ではパラメータとして受け入れられません。書式の未確定解釈に必要な特定のデータ項目を明示的に定義する NLS パラメータのみが受け入れられます。NLS_DATE_FORMAT も、次の理由によってパラメータとして受け入れられません。

NLS パラメータを TO_CHAR、TO_NUMBER、または TO_DATE に指定する場合、書式マスクも第 2 パラメータとして指定する必要があります。たとえば、次の指定は有効です。

```
TO_CHAR (hiredate, 'DD/MON/YYYY', 'nls_date_language = French')
```

次の指定は無効です。

```
TO_CHAR (hiredate, 'nls_date_language = French')
TO_CHAR (hiredate, 'nls_date_language = French',
   'DD/MON/YY')
```

この制約では、NLS パラメータが TO_CHAR 関数または TO_DATE 関数にある場合は、常に、日付書式を指定しなければなりません。その結果、NLS_DATE_FORMAT は、これらの関数に有効な NLS パラメータではなくなります。

CONVERT 関数

SQL 関数 CONVERT は、キャラクタ・セット間で文字データを変換します。

CONVERT 関数は、あるキャラクタ・セットの文字列のバイナリ表現を別のキャラクタ・セットのバイナリ表現に変換します。この関数は、データベースのキャラクタ・セットおよびクライアントのキャラクタ・セット間の変換について前述したものと同じ技法を使用します。したがって、この関数では置換文字を使用するため、同じ制限があります。

CONVERT 関数をストアド・プロシージャで使用すると、そのストアド・プロシージャは、クライアントのキャラクタ・セットを使用せずにサーバーのキャラクタ・セットを使用して実行されます。これによって、変換された文字の最後が切り捨てられることがあります。

CONVERT の構文は、次のとおりです。

この *src_char_set* はソースのキャラクタ・セットであり、*dest_char_set* は変換先のキャラクタ・セットです。

異なるキャラクタ・セットを使用するクライアント / サーバー環境で変換を実行するには、CONVERT 文のかわりに TRANSLATE (...USING...) 文を使用します。これによって、クライアントのキャラクタ・セットへの変換時に、TRANSLATE 文の結果であるサーバーのキャラクタ・セットが正しく認識されます。

CONVERT の詳細は、『Oracle8*i* SQL リファレンス』を参照してください。

キャラクタ・セットの SQL 関数

キャラクタ・セット ID 番号とキャラクタ・セット名の間の変換のために、2 つの SQL 関数 NLS_CHARSET_NAME および NLS_CHARSET_ID が用意されています。これらの関数は、OCI を介して変数をバインドするためのキャラクタ・セット ID 番号を判断する必要のあるプログラムで使用されます。

NLS_CHARSET_DECL_LEN 関数は、NCHAR 列の宣言の長さ（文字数）を戻します。

これらの関数の詳細は、『Oracle8*i* SQL リファレンス』を参照してください。

キャラクタ・セット番号からキャラクタ・セット名への変換

NLS_CHARSET_NAME(*n*) 関数は、ID 番号 *n* に対応するキャラクタ・セットの名前を戻します。*n* がキャラクタ・セット ID として認識されない値である場合、関数は NULL を戻します。

キャラクタ・セット名からキャラクタ・セット番号への変換

NLS_CHARSET_ID(*TEXT*) は、*TEXT* によって指定された名前に対応するキャラクタ・セット ID を戻します。*TEXT* は、キャラクタ・セット名であるランタイム VARCHAR2 の量として定義されます。*TEXT* の値は、データベース・キャラクタ・セットや各国キャラクタ・セット以外のキャラクタ・セットに変換される NLSRTL 名の値とすることができます。

値 CHAR_CS を *TEXT* に入力すると、関数はサーバーのデータベース・キャラクタ・セット ID を戻します。値 NCHAR_CS を *TEXT* に入力すると、関数はサーバーの各国キャラクタ・セット ID を戻します。*TEXT* が名前として認識されない場合、関数は NULL を戻します。*TEXT* の値は、すべて大文字で入力してください。

NCHAR 列の長さを戻す

NLS_CHARSET_DECL_LEN(*BYTECNT, CSID*) は、NCHAR 列の宣言の長さ（文字数）を戻します。*BYTECNT* 引数は NCHAR 列のバイト単位の長さです。*CSID* 引数は NCHAR 列のキャラクタ・セット ID です。

NLSSORT 関数

NLSSORT 関数は、文字列を、言語ソート方法で使用されるのと同等なソート文字列に置換します。バイナリ・ソートの場合、ソート文字列は、入力文字列と同じです。結果的に生じた文字列をソートすると希望するソート基準になるように選択された他の 2 進値に、それぞれの文字列を置換するという方法で、言語ソート技法は動作します。言語ソートが使用されているときは、NLSSORT は元の文字列に置き換わる 2 進値を戻します。

SQL 文にある ORDER BY 句は、セッション・パラメータの NLS_SORT によって決定されますが、次に示す例のように、明示的に NLSSORT() 関数を使用することによってその値を変更できます。

```
ALTER SESSION SET NLS_SORT = GERMAN;
SELECT
FROM
ORDER BY col1;
```

前述の例ではドイツ語によるソートを使用していますが、次の例ではフランス語でソートしています。

```
ALTER SESSION SET NLS_SORT = GERMAN;
SELECT
FROM
ORDER BY NLSSORT(col1, 'NLS_SORT = FRENCH');
```

WHERE 句は、通常、言語上の比較ではなくバイナリ比較を使用します。ただし、この比較は次の 2 通りの方法で変更することができます。

1. WHERE 句で NLSSORT() 関数を使用する。

```
SELECT
FROM
WHERE NLSSORT(col1, 'NLS_SORT = FRENCH')>
              NLSSORT(col2, 'NLS_SORT = FRENCH');
```

2. WHERE 句に NLS_SORT セッション・パラメータが使用されている場合、セッション・パラメータ NLS_COMP に ASCII を設定する。

```
ALTER SESSION SET NLS_COMP = ASCII;
```

NLSSORT の構文

NLSSORT は、次の 4 通りの方法で使用できます。

- NLSSORT()—NLS_SORT パラメータに従う
- NLSSORT(" ", "NLS_SORT=xxxx")
- NLSSORT(" ", "NLS_LANG= xxxx")
- NLSSORT(" ", "NLS_LANGUAGE=xxxx")

WHERE 句の文字列の比較

NLSSORT によって、アプリケーションは、アルファベットの規則に従う文字列のマッチングを実行できます。通常、WHERE 句の文字列は、文字の 2 進値を使用して比較されます。ある文字が、データベースのキャラクタ・セット内で、別の文字よりも高い 2 進値を持っている場合、その文字のほうがより大きい文字です。文字の 2 進値に基づいた文字順が言語のアルファベット順と一致しない場合があるので、このような比較では、アルファベットの規則に従わないことがあります。たとえば、列（COL1）に ISO 8859/1 8 ビット・キャラクタ・セットの値 ABC、ABZ、BCD および ÄBC がある場合、次の問合せでは、

```
SELECT col1 FROM tab1 WHERE col1 > 'B'
```

BCD および ÄBC の両方が戻ります。これは、Ä の数値は B よりも高位であるためです。しかし、ドイツ語のアルファベットでは、Ä は B より前になります。このような規則は、同じ文字が使用されるときでも、言語によって異なります。スウェーデン語では、Ä は Z の後にソートされます。次に示すように WHERE 句で NLSSORT を使用して、言語上の比較を実行できます。

```
WHERE NLSSORT(col) comparison_operator NLSSORT(comparison_string)
```

NLSSORT は、比較演算子の両側に指定しなければなりません。たとえば、次のように指定します。

```
SELECT col1 FROM tab1 WHERE NLSSORT(col1) > NLSSORT('B')
```

ドイツ語の言語ソートを使用していると、上の問合せでは Ä で始まる文字列は戻されません。これは、ドイツ語では、アルファベット順で Ä は B より前になるためです。スウェーデン語では Ä は Z より後になるため、スウェーデン語の言語ソートを使用していると、この文字で始まる名前が戻されます。

NLS_COMP

通常、WHERE 句での比較はバイナリで行われます。言語上の比較を実行するには、NLSSORT 関数を使用しなければなりません。NLSSORT 関数の使用は、必要な言語ソートがすでに NLS_SORT セッションのパラメータに指定されている場合など、特に面倒に感じられます。このような場合、NLS_COMP を使用すると、比較が NLS_SORT セッション・パラメータに従って言語的に行われるように指示できます。セッションの変更で、次のように入力します。

```
SQL> ALTER SESSION SET NLS_COMP = ASCII;
```

WHERE 句での比較が必ずバイナリで行われるように指定するには、次のように入力します。

```
SQL> ALTER SESSION SET NLS_COMP = BINARY;
```

NLS_COMP に ASCII が設定されている場合、言語上の順序が希望されている列には言語索引が存在しなければなりません。

言語索引を使用可能にするには、次の構文を使用します。

```
SQL> CREATE INDEX i ON t(NLSSORT(col, 'NLSSORT=FRENCH'));
```

パーティション表および索引

DDL および DML 用のパーティション VALUES LESS THAN（より小さい）照合での文字列の比較は、常に BINARY 順序に従います。

ORDER BY 句の制御

言語ソート基準が使用されていると、ORDER BY 句内のそれぞれの文字項目に対して NLSSORT が暗黙に使用されます。その結果、ORDER BY のソート方法（言語またはバイナリ）は、アプリケーション側で意識することはありません。ただし、NLSSORT 関数を、ORDER BY 項目内の文字項目について明示的に指定すると、暗黙の NLSSORT は実行されません。

つまり、NLSSORT の言語的置換は 1 回だけ適用され、2 回適用されることはありません。デフォルトのソート方法が言語ソートであるとき、NLSSORT 関数は、通常 ORDER BY 句では必要ありません。しかし、デフォルトのソート方法が BINARY であるときは、問合せは次のようになります。

```
SELECT ename FROM emp
ORDER BY ename
```

この問合せではバイナリ・ソートを使用します。次の問合せによって、ドイツ語の言語ソートを取得できます。

```
SELECT ename FROM emp
ORDER BY NLSSORT(ename, 'NLS_SORT = GERMAN')
```

固定幅マルチバイト・キャラクタ・セットのパターン一致文字

パターン一致を使用した文字列比較では、LIKE 演算子を使用します。この構文では、アンダースコア（_）およびパーセント符号（%）の 2 つの特殊パターン一致文字を使用する必要があります。

表 4-1 アンダースコア、パーセント符号および埋込み文字のコード化

キャラクタ・セット	使用するコード・ポイントの値		
	アンダースコア	パーセント符号	埋込み文字（空白）
JA16SJISFIXED	0x8151	0x8193	0x8140
JA16EUCFIXED	0xa1b2	0xa1f3	0xa1a1
JA16DBCSFIXED	0x426d	0x426c	0x4040
ZHT32TRISFIXED	0x8eb1a1df	0x8eb1a1a5	0x8ebla1a0

時刻 / 日付 / カレンダの書式

関連規則に従って日付および数値を書式化するために、いくつかの書式マスクが TO_CHAR 関数、TO_DATE 関数および TO_NUMBER 関数で提供されます。

注意 : TO_NUMBER 関数でも書式マスクを使用できます。

日付書式

書式要素 RM（Roman Month）は、ローマ数字で月を戻します。RM または rm を使用して、大文字または小文字のいずれかを指定できます。たとえば、1998 年 9 月 7 日という日付の場合、「DD-rm-YYYY」では「07-ix-1998」が戻り、「DD-RM-YYYY」では「07-IX-1998」が戻ります。

書式マスク MON および DY は、月および曜日の略称が 3 文字の長さでない場合には、その略称を明示的にサポートします。たとえば、フランス語の「Lundi」および「Mardi」の場合、それぞれの略称「Lu」および「Ma」を指定できます。

週および日の番号の規則

書式マスク WW で戻される週番号は、アルゴリズム *int((day-ijan1)/7)* に従って算出されます。この週番号のアルゴリズムは、ISO 規格（2015、1992-06-15）に従っていません。

ISO 規格をサポートするために、ISO 週番号を戻す書式要素 IW があります。さらに、書式要素 Y、YY、YYY および YYYY の動作と等価の書式要素 I、IY、IYY および IYYY は、ISO 週番号に関連する年を戻します。

ISO 規格では、ISO 週番号に関連する年は、カレンダ年と異なることがあります。たとえば、1988 年 1 月 1 日は、1987 年の ISO 週番号 53 になります。週は、常に月曜日に始まって日曜日に終わります。

- 1 月 1 日が金曜日、土曜日または日曜日であれば、1 月 1 日を含む週は、この週の大半の日が前年に属するため、前年の最後の週になります。
- 1 月 1 日が月曜日、火曜日、水曜日または木曜日であれば、1 月 1 日を含む週は、この週の大半の日が新年に属するため、新年の最初の週になります。

たとえば、1991 年 1 月 1 日は火曜日なので、1990 年 12 月 31 日の月曜日から 1991 年 1 月 6 日の日曜日までが第 1 週になります。したがって、1990 年 12 月 31 日の ISO 週番号および年は、1 および 1991 となります。ISO 週番号を取得するには、週番号に書式マスク「IW」を使用し、年に「IY」書式のいずれかを使用します。

数値書式

数値を書式化するために、その他の書式要素がいくつか提供されています。

- D（Decimal）は、小数点文字を戻す。
- G（Group）は、グループ・セパレータを戻す。
- L（Local Currency）は、各国通貨記号を戻す。
- C（International Currency）は、国際通貨記号を戻す。
- RN（Roman Numeral）は、数値を、それに等価なローマ数字として戻す。

ローマ数字の場合、RN または rn をそれぞれ使用して、大文字または小文字のいずれかを指定できます。変換される数値は、1 ～ 3999 の範囲の整数でなければなりません。

日付マスクと数値マスクの使用方法の詳細は、『Oracle8*i* SQL リファレンス』を参照してください。

その他のトピック

連結演算子

データベース・キャラクタ・セットが垂直バー（"|"）を各国文字に置換すると、連結演算子（ASCII 124）を使用する SQL 文はすべて失敗します。たとえば、連結を使用する再帰的 SQL 文を生成するため、プロシージャの作成は失敗します。データベース・キャラクタ・セットに D7DEC、F7DEC、SF7ASCII などの 7 ビット置換キャラクタ・セットを使用する場合、垂直バーが連結演算子として解釈されるため、垂直バーを置換する各国文字はオブジェクト名に使用できません。

ユーザー側では、データベース・キャラクタ・セットが同じか、または互換性がある場合（つまり、両方のキャラクタ・セットが垂直バーを同じ各国文字に置換する場合）、7 ビットの置換キャラクタ・セットを使用できます。

5
OCI プログラミング

この章では、OCI プログラミングで役立つ次の情報について説明します。

- NLS 言語情報の取出し
- 文字列操作
- 文字の分類
- キャラクタ・セット変換
- メッセージ・メカニズム

NLS 言語情報の取出し

Oracle のロケールは、言語、地域およびキャラクタ・セットの定義で構成されます。ロケールによって、日付、時刻、数字および通貨の書式と、ネイティブの曜日名、月名などの規則が定義されます。国際化されたアプリケーションは、ユーザーのロケール設定と文化的規則に従って動作します。たとえば、ドイツ語のロケール設定では、曜日名および月名はドイツ語で表示されます。

OCINlsGetInfo()

環境ハンドルを使用すると、次の情報を取り出すことができます。

- 週の最初の曜日（変換後）
- 週の曜日の略称（変換後）
- 月名（変換後）
- 月の略称（変換後）
- Yes / No
- AM / PM
- AD / BC
- 数値書式
- 借方 / 貸方
- 日付書式
- 通貨書式
- デフォルトの言語
- デフォルトの地域
- デフォルトのキャラクタ・セット
- デフォルトの言語ソート
- デフォルトのカレンダ

OCINlsGetInfo

構文

```
sword OCINlsGetInfo(dvoid *hndl, OCIError *errhp, text *buf, size_t buflen, ub2
item)
```

備考

この関数は、item で指定した言語情報を OCI 環境ハンドルまたはユーザー・セッション・ハンドルである hndl から生成し、buflen をサイズの制限値として buf によって示された配列に格納します。

戻り値

OCI_SUCCESS、OCI_INVALID_HANDLE または OCI_ERROR。

表 5-1 OCINlsGetInfo のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	オブジェクト・モードで初期化された OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
errhp（IN/OUT）	OCI エラー・ハンドル。エラーがある場合は、errhp に記録され、NULL ポインタが戻されます。診断情報は OCIErrorGet() をコールすると取得できます。
buf（OUT）	宛先バッファへのポインタ。
buflen（IN）	宛先バッファのサイズ。それぞれの情報の最大長は、OCI_NLS_MAXBUFSZ バイトです。
item（IN）	OCI 環境ハンドルで取得する項目。次のいずれかの値を指定できます。
	OCI_NLS_DAYNAME1: Monday のネイティブの名前。
	OCI_NLS_DAYNAME2: Tuesday のネイティブの名前。
	OCI_NLS_DAYNAME3: Wednesday のネイティブの名前。
	OCI_NLS_DAYNAME4: Thursday のネイティブの名前。
	OCI_NLS_DAYNAME5: Friday のネイティブの名前。
	OCI_NLS_DAYNAME6: Saturday のネイティブの名前。
	OCI_NLS_DAYNAME7: Sunday のネイティブの名前。
	OCI_NLS_ABDAYNAME1: Monday のネイティブの略称。
	OCI_NLS_ABDAYNAME2: Tuesday のネイティブの略称。
	OCI_NLS_ABDAYNAME3: Wednesday のネイティブの略称。
	OCI_NLS_ABDAYNAME4: Thursday のネイティブの略称。
	OCI_NLS_ABDAYNAME5: Friday のネイティブの略称。
	OCI_NLS_ABDAYNAME6: Saturday のネイティブの略称。
	OCI_NLS_ABDAYNAME7: Sunday のネイティブの略称。

表 5-1 OCINlsGetInfo のキーワード / パラメータ

キーワード / パラメータ	意味
	OCI_NLS_MONTHNAME1: January のネイティブの名前。
	OCI_NLS_MONTHNAME2: February のネイティブの名前。
	OCI_NLS_MONTHNAME3: March のネイティブの名前。
	OCI_NLS_MONTHNAME4: April のネイティブの名前。
	OCI_NLS_MONTHNAME5: May のネイティブの名前。
	OCI_NLS_MONTHNAME6: June のネイティブの名前。
	OCI_NLS_MONTHNAME7: July のネイティブの名前。
	OCI_NLS_MONTHNAME8: August のネイティブの名前。
	OCI_NLS_MONTHNAME9: September のネイティブの名前。
	OCI_NLS_MONTHNAME10: October のネイティブの名前。
	OCI_NLS_MONTHNAME11: November のネイティブの名前。
	OCI_NLS_MONTHNAME12: December のネイティブの名前。
	OCI_NLS_ABMONTHNAME1: January のネイティブの略称。
	OCI_NLS_ABMONTHNAME2: February のネイティブの略称。
	OCI_NLS_ABMONTHNAME3: March のネイティブの略称。
	OCI_NLS_ABMONTHNAME4: April のネイティブの略称。
	OCI_NLS_ABMONTHNAME5: May のネイティブの略称。
	OCI_NLS_ABMONTHNAME6: June のネイティブの略称。
	OCI_NLS_ABMONTHNAME7: July のネイティブの略称。
	OCI_NLS_ABMONTHNAME8: August のネイティブの略称。
	OCI_NLS_ABMONTHNAME9: September のネイティブの略称。
	OCI_NLS_ABMONTHNAME10: October のネイティブの略称。
	OCI_NLS_ABMONTHNAME11: November のネイティブの略称。
	OCI_NLS_ABMONTHNAME12: December のネイティブの略称。

表 5-1 OCINlsGetInfo のキーワード / パラメータ

キーワード / パラメータ	意味
	OCI_NLS_YES: 肯定的な応答についてのネイティブの文字列。
	OCI_NLS_NO: ネイティブの否定的な応答。
	OCI_NLS_AM: AM に相当するネイティブの文字列。
	OCI_NLS_PM: PM に相当するネイティブの文字列。
	OCI_NLS_AD: AD に相当するネイティブの文字列。
	OCI_NLS_BC: BC に相当するネイティブの文字列。
	OCI_NLS_DECIMAL: 小数点文字。
	OCI_NLS_GROUP: グループ・セパレータ。
	OCI_NLS_DEBIT: ネイティブの借方記号。
	OCI_NLS_CREDIT: ネイティブの貸方記号。
	OCI_NLS_DATEFORMAT: Oracle の日付書式。
	OCI_NLS_INT_CURRENCY: 国際通貨記号。
	OCI_NLS_LOC_CURRENCY: 各国通貨記号。
	OCI_NLS_LANGUAGE: 言語名。
	OCI_NLS_ABLANGUAGE: 言語名の略称。
	OCI_NLS_TERRITORY: 地域名。
	OCI_NLS_CHARACTER_SET: キャラクタ・セット名。
	OCI_NLS_LINGUISTIC_NAME: 言語学上の名前。
	OCI_NLS_CALENDAR: カレンダ名。

OCI_Nls_MaxBufSz

OCINlsGetInfo() をコールする場合、特定の言語について戻された情報を格納するためのバッファを割り当てておく必要があります。バッファ・サイズは、どの項目に対して問合せを行ったか、どのコード化を使用して情報を格納するかによって異なります。JA16SJIS コードを使用して日本語で「January」を格納するために必要なバイト数を、開発者が意識する必要はありません。つまり、バッファを割り当てる場合は、OCI_NLS_MAXBUFSZ を使用します。OCI_NLS_MAXBUFSZ は、OCINlsGetInfo() が戻す最大の項目を保持するのに十分な大きさであることが保証されています。

これによって、OCINlsGetInfo() によって戻される最大の項目をバッファ内に確実に収めることができます。

詳細は、『Oracle8*i* コール・インタフェース・プログラマーズ・ガイド』および『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

NLS 言語情報の取出しのサンプル・コード

次に、情報を取り出しそのエラーをチェックする簡単な事例を示します。

```
sword MyPrintLinguisticName(envhp, errhp)
OCIEnv   *envhp;
OCIError *errhp;
{
  text  infoBuf[OCI_NLS_MAXBUFSZ];
  sword ret;

  ret = OCINlsGetInfo(envhp,                          /* environment handle */
                      errhp,                               /* error handle */
                      infoBuf,                        /* destination buffer */
                      (size_t) OCI_NLS_MAXBUFSZ,             /* buffer size */
                      (ub2) OCI_NLS_LINGUISTIC);                    /* item */

  if (ret != OCI_SUCCESS)
  {
    checkerr(errhp, ret, OCI_HTYPE_ERROR);
    ret = OCI_ERROR;
  }
  else
  {
    printf("NLS linguistic: %s\n", infoBuf);
   }
  return(ret);
}
```

文字列操作

文字列操作では、マルチバイト文字列とワイド・キャラクタ文字列の 2 種類のデータ構造がサポートされています。マルチバイト文字列は、Oracle のネイティブ・キャラクタ・セットでコード化されており、マルチバイト文字列を処理する関数は、その文字列を 1 つの単位とみなします。ワイド・キャラクタ文字列の wchar 関数は、文字列をより柔軟に操作し、文字ベースおよび文字列ベースの操作をサポートします。

ワイド・キャラクタのデータ型は Oracle 固有のデータ型です。ANSI / ISO の C 規格で定義されている `wchar_t` と混同しないでください。Oracle のワイド・キャラクタは、すべてのプラットフォームで常に 4 バイトですが、`wchar_t` は、インプリメンテーションおよびプラットフォームに依存します。Oracle のワイド・キャラクタは、マルチバイト文字を正規化し固定幅コードを簡単に処理できるようにすることがねらいです。これによって、Oracle の

ワイド・キャラクタとネイティブ・キャラクタ・セットの間のラウンドトリップ変換が保証されます。

文字列操作は、次のように分類できます。

- マルチバイト文字とワイド・キャラクタ間の文字列の変換
- 文字の分類
- 文字の変換
- 表示長の計算
- 比較、連結、検索などの一般の文字列操作

表 5-2 OCI 文字列操作コール

関数コール	説明
OCIMultiByteToWideChar()	NULL で終了する文字列全体を wchar 書式に変換します。
OCIMultiByteInSizeToWideChar()	文字列の一部を wchar 書式に変換します。
OCIWideCharToMultiByte()	NULL で終了するワイド・キャラクタ文字列全体をマルチバイト文字列に変換します。
OCIWideCharInSizeToMultiByte()	ワイド・キャラクタ文字列の一部をマルチバイト書式に変換します。
OCIWideCharToLower()	指定されたロケールに小文字の文字マッピングがある場合、そのワイド・キャラクタを小文字で戻します。小文字の文字マッピングがない場合、そのワイド・キャラクタを戻します。
OCIWideCharToUpper()	指定されたロケールに大文字の文字マッピングがある場合、そのワイド・キャラクタを大文字で戻します。大文字の文字マッピングがない場合、そのワイド・キャラクタを戻します。
OCIWideCharStrcmp()	2 つのワイド・キャラクタ文字列をバイナリ、言語上の方法または大文字と小文字を区別しない方法で比較します。
OCIWideCharStrncmp()	OCIWideCharStrcmp() と似ているが、多少異なります。
OCIWideCharStrcat()	wrcstr で示された文字列のコピーを追加します。その後、結果文字列の文字数を戻します。
OCIWideCharStrchr()	wstr で示された文字列の中で、最初に出現する wc を検索します。正常終了した場合は、wchar へのポインタを戻します。
OCIWideCharStrcpy()	wsrcstr で示された wchar 文字列を wdststr で示された配列にコピーします。その後、コピーされた文字数を戻します。
OCIWideCharStrlen()	wstr で示された wchar 文字列にある文字数を計算し、その数を戻します。
OCIWideCharStrncat()	wsrcstr で示された文字列のコピーを追加します。その後、結果文字列の文字数を戻します。最大 n 文字が追加されます。

表 5-2 OCI 文字列操作コール

関数コール	説明
OCIWideCharStrncpy()	wsrcstr で示された wchar 文字列を wdststr で示された配列にコピーします。その後、コピーされた文字数を戻します。最大 n 文字が配列にコピーされます。
OCIWideCharStrrchr()	wstr で示された文字列の中で、最後に出現する wc を検索します。
OCIWideCharStrCase Conversion()	wsrcstr で示されたワイド・キャラクタ文字列をフラグで指定された大文字または小文字に変換し、その結果を wdststr で示された配列にコピーします。
OCIWideCharDisplayLength()	wc を表示するのに必要な列位置の数を決定します。
OCIWideCharMultibyteLength()	wc をマルチバイトでコード化する場合に必要なバイト数を決定します。
OCIMultiByteStrcmp()	2 つのマルチバイト文字列をバイナリ、言語上の方法または大文字と小文字を区別しない方法で比較します。
OCIMultiByteStrncmp()	2 つのマルチバイト文字列をバイナリ、言語上の方法または大文字と小文字を区別しない方法で比較します。str1 の先頭から len1 バイトと str2 の先頭から len2 バイトが比較されます。
OCIMultiByteStrcat()	srcstr で示されたマルチバイト文字列のコピーを追加します。
OCIMultiByteStrcpy()	srcstr で示されたマルチバイト文字列を dststr で示された配列にコピーします。その後、コピーされたバイト数を戻します。
OCIMultiByteStrlen()	str で示されたマルチバイト文字列にあるバイト数を計算し、その数を戻します。
OCIMultiByteStrncat()	srcstr で示されたマルチバイト文字列のコピーを追加します。最大 n バイトが srcstr から dststr に追加されます。
OCIMultiByteStrncpy()	srcstr で示されたマルチバイト文字列を dststr で示された配列にコピーします。その後、コピーされたバイト数を戻します。最大 n バイトが srcstr で示された配列から dststr で示された配列にコピーされます。
OCIMultiByteStrnDisplayLength()	n バイトの範囲内のすべての文字が占有する表示位置の数を戻します。
OCIMultiByteStrCase Conversion()	文字列の一部をあるキャラクタ・セットから別のキャラクタ・セットに変換します。

OCIMultiByteToWideChar

構文

```
sword OCIMultiByteToWideChar(dvoid *hndl, OCIWchar *dst, CONST text *src, size_t
*rsize);
```

備考

このルーチンは、NULL で終了する文字列全体を wchar 書式に変換します。wchar アウトプット・バッファは、NULL で終了します。

戻り値

OCI_SUCCESS、OCI_INVALID_HANDLE または OCI_ERROR。

表 5-3 OCIMultiByteToWideChar のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	文字列のキャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
dst（OUT）	wchar の宛先バッファ。
src（IN）	変換するソース文字列。
rsize（OUT）	変換された文字数（NULL 終了記号を含む）。 NULL ポインタの場合は、戻り値がないことを意味します。

OCIMultiByteInSizeToWideChar

構文

```
sword OCIMultiByteInSizeToWideChar(dvoid *hndl, OCIWchar *dst, size_t dstsz, CONST
text *src, size_t srcsz, size_t *rsize)
```

備考

このルーチンは、文字列の一部を wchar 書式に変換します。アウトプット・バッファ・サイズまたは入力バッファ・サイズに到達するまで、あるいはソース文字列が NULL 終了文字に到達するまで、すべての文字を変換します。アウトプット・バッファは、空き領域があれば NULL で終了します。dstsz がゼロの場合、この関数は変換された文字列に必要な文字数（NULL 終了記号を含まない）のみを戻します。

戻り値

OCI_SUCCESS、OCI_INVALID_HANDLE または OCI_ERROR。

表 5-4 OCIMultiByteInSizeToWideChar のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	文字列のキャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
dst（OUT）	wchar 用の宛先バッファへのポインタ。dstsz がゼロの場合、NULL ポインタになります。

表 5-4 OCIMultiByteInSizeToWideChar のキーワード / パラメータ

キーワード / パラメータ	意味
dstsz（IN）	宛先バッファ・サイズ（文字単位）。ゼロの場合、この関数は変換に必要な文字数のみを戻します。
src（IN）	変換するソース文字列。
srcsz（IN）	ソース文字列の長さ（バイト単位）。
rsize（OUT）	宛先バッファに書き込まれた文字数、または、dstsz がゼロの場合は、その文字列を変換するために必要な文字数。NULL ポインタの場合は、戻り値がないことを意味します。

OCIWideCharToMultiByte

構文

```
sword OCIWideCharToMultiByte(dvoid *hndl, text *dst, CONST OCIWchar *src, size_t
*rsize)
```

備考

このルーチンは、NULL で終了するワイド・キャラクタ文字列全体をマルチバイト文字列に変換します。アウトプット・バッファは、NULL で終了します。

戻り値

OCI_SUCCESS、OCI_INVALID_HANDLE または OCI_ERROR。

表 5-5 OCIWideCharToMultiByte のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	文字列のキャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
dst（OUT）	マルチバイト文字列の宛先バッファ。
src（IN）	変換するソース wchar 文字列。
srcsz（IN）	ソース文字列の長さ（バイト単位）。
rsize（OUT）	宛先バッファに書き込まれた文字数。NULL ポインタの場合は、戻り値がないことを意味します。

OCIWideCharInSizeToMultiByte

```
sword OCIWideCharInSizeToMultiByte(dvoid *hndl, text *dst, size_t dstsz, CONST
OCIWchar *src, size_t srcsz, size_t *rsize)
```

備考

このルーチンは、wchar 文字列の一部をマルチバイト書式に変換します。アウトプット・バッファ・サイズまたは入力バッファ・サイズに到達するまで、あるいはソース文字列が NULL 終了文字に到達するまで、すべての文字を変換します。アウトプット・バッファは、空き領域があれば NULL で終了します。dstsz がゼロの場合、この関数は変換された文字列を格納するために必要なサイズ（NULL 終了記号を含まない）をバイト単位で戻します。

戻り値

OCI_SUCCESS、OCI_INVALID_HANDLE または OCI_ERROR。

表 5-6 OCIWideCharInSizeToMultiByte のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	文字列のキャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
dst（OUT）	マルチバイトの宛先バッファ。dstsz がゼロの場合、NULL ポインタになります。
dstsz（IN）	宛先バッファ・サイズ（バイト単位）。ゼロの場合、変換された文字列に必要なサイズ（バイト単位）のみを戻します。
src（IN）	変換するソース wchar 文字列。
srcsz（IN）	ソース文字列の長さ（文字単位）。
rsize（OUT）	宛先バッファに書き込まれたバイト数、または、dstsz がゼロの場合、変換された文字列を格納するために必要なバイト数。NULL ポインタの場合は、戻り値がないことを意味します。

OCIWideCharToLower

構文

```
OCIWchar OCIWideCharToLower(dvoid *hndl, OCIWchar wc)
```

備考

指定されたロケールに wc に対する小文字の文字マッピングがある場合、その wchar を小文字で戻します。小文字の文字マッピングがない場合は、wc をそのまま戻します。

戻り値
wchar。

表 5-7 OCIWideCharToLower のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	小文字マッピングに対する wchar。

OCIWideCharToUpper

構文
```
OCIWchar OCIWideCharToUpper(dvoid *hndl, OCIWchar wc)
```

備考
指定されたロケールに wc に対する大文字の文字マッピングがある場合、その wchar を大文字で戻します。大文字の文字マッピングがない場合は、wc をそのまま戻します。

戻り値
wchar。

表 5-8 OCIWideCharToUpper のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	大文字マッピングに対する wchar。

OCIWideCharStrcmp

構文
```
int OCIWideCharStrcmp(dvoid *hndl, CONST OCIWchar *wstr1, CONST OCIWchar *wstr2, int
flag)
```

備考
2 つの wchar 文字列を、（wchar のコード値に基づいた）バイナリ、言語上の方法、または大文字と小文字を区別しない方法で比較します。

戻り値
- wstr = wstr2 の場合、0。

- wstr > wstr2 の場合、正の値。
- wstr < wstr2 の場合、負の値。

表 5-9 OCIWideCharStrcmp のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wstr1（IN）	NULL で終了する wchar 文字列へのポインタ。
wstr2（IN）	NULL で終了する wchar 文字列へのポインタ。
flag（IN）	比較方法を決定するために使用されるフラグ。次のいずれかの値をとることができます。 ■ OCI_NLS_BINARY: バイナリでの比較。これは、デフォルト値です。 ■ OCI_NLS_LANGUISTIC: ロケールで指定された言語上の比較。 大文字と小文字を区別せずに比較するには、このフラグに OCI_NLS_CASE_INSENSITIVE と OR された値を指定します。

OCIWideCharStrncmp

構文

```
int OCIWideCharStrncmp(dvoid *hndl, CONST OCIWchar *wstr1, size_t len1, CONST
OCIWchar *wstr2, size_t len2, int flag)
```

備考

この関数は、wstr1 の先頭 len1 文字と wstr2 の先頭 len2 文字のみが比較されることを除けば、OCIWideCharStrcmp() と似ています。NULL 終了文字も比較の対象です。

戻り値

- wstr = wstr2 の場合、0。
- wstr > wstr2 の場合、正の値。
- wstr < wstr2 の場合、負の値。

表 5-10 OCIWideCharStrncmp のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wstr1（IN）	最初の wchar 文字列へのポインタ。
len1（IN）	比較する最初の文字列の長さ。
wstr2（IN）	2 番目の wchar 文字列へのポインタ。
len2（IN）	比較する 2 番目の文字列の長さ。
flag（IN）	比較方法を決定するために使用されるフラグ。次のいずれかの値をとることができます。 ■ OCI_NLS_BINARY: バイナリでの比較。これは、デフォルト値です。 ■ OCI_NLS_LANGUISTIC: ロケールで指定された言語上の比較。 大文字と小文字を区別せずに比較するには、このフラグに OCI_NLS_CASE_INSENSITIVE と OR された値を指定します。

OCIWideCharStrcat

構文

```
size_t OCIWideCharStrcat(dvoid *hndl, OCIWchar *wdststr, CONST OCIWchar *wsrcstr)
```

備考

この関数は、wsrcstr で示された wchar 文字列のコピーを、wdststr で示された wchar 文字列の最後に NULL 終了文字も含めて追加します。

戻り値

結果文字列の文字数（最後の NULL 終了文字を含まない）。

表 5-11 OCIWideCharStrcat のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wdststr（IN/OUT）	追加先の wchar 文字列へのポインタ。
wsrcstr（IN）	追加するソース wchar 文字列へのポインタ。

OCIWideCharStrchr

構文

```
OCIWchar *OCIWideCharStrchr(dvoid *hndl, CONST OCIWchar *wstr, OCIWchar wc)
```

備考

この関数は、wstr で示された wchar 文字列の中で、最初に出現する wc を検索します。

戻り値

正常終了した場合は wchar へのポインタ、それ以外の場合は NULL ポインタ。

表 5-12 OCIWideCharStrchr のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wstr（IN）	検索する wchar 文字列へのポインタ。
wc（IN）	検索する wchar。

OCIWideCharStrcpy

構文

```
size_t OCIWideCharStrcpy(dvoid *hndl, OCIWchar *wdststr, CONST OCIWchar *wsrcstr)
```

備考

この関数は、wsrcstr で示された wchar 文字列のコピーを、wdststr で示された配列に NULL 終了文字も含めて追加します。

戻り値

コピーされた文字数（最後の NULL 終了文字を含まない）。

表 5-13 OCIWideCharStrcpy のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wdststr（OUT）	追加先の wchar バッファへのポインタ。
wsrcstr（IN）	ソース wchar 文字列へのポインタ。

OCIWideCharStrlen

構文

```
size_t OCIWideCharStrlen(dvoid *hndl, CONST OCIWchar *wstr)
```

備考

この関数は、wstr で示された wchar 文字列の文字数（NULL 終了文字を含まない）を計算し、その数を戻します。

戻り値

文字数（最後の NULL 終了文字を含まない）。

表 5-14 OCIWideCharStrlen のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wstr（IN）	ソース wchar 文字列へのポインタ。

OCIWideCharStrncat

構文

```
size_t OCIWideCharStrncat(dvoid *hndl, OCIWchar *wdststr, CONST OCIWchar *wsrcstr,
size_t n)
```

備考

この関数は、最大 n 文字が wsrcstr から wdststr に追加されることを除けば、OCIWideCharStrcat() に似ています。wsrcstr に NULL 終了文字があると、そこで追加処理は停止します。wdststr は、NULL で終了します。

戻り値
結果文字列の文字数（最後の NULL 終了文字を含まない）。

表 5-15 OCIWideCharStrncat のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wdststr（IN/OUT）	追加先の wchar 文字列へのポインタ。
wsrcstr（IN）	追加するソース wchar 文字列へのポインタ。
n（IN）	wsrcstr から追加する文字数。

OCIWideCharStrncpy

構文
```
size_t OCIWideCharStrncpy(dvoid *hndl, OCIWchar *wdststr, CONST OCIWchar *wsrcstr,
size_t n)
```

備考
この関数は、最大 n 文字が wsrcstr で示された配列から wdststr で示された配列に追加されることを除けば、OCIWideCharStrcpy() に似ています。wsrcstr に NULL 終了文字があると、そこでコピー処理は停止します。結果文字列は NULL で終了します。

戻り値
コピーされた文字数（最後の NULL 終了文字を含まない）。

表 5-16 OCIWideCharStrncpy のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wdststr（OUT）	追加先の wchar バッファへのポインタ。
wsrcstr（IN）	ソース wchar 文字列へのポインタ。
n（IN）	wsrcstr からコピーする文字数。

OCIWideCharStrrchr

構文

```
OCIWchar *OCIWideCharStrrchr(dvoid *hndl, CONST OCIWchar *wstr, OCIWchar wc)
```

備考

この関数は、wstr で示された wchar 文字列の中で、最初に出現する wc を検索します。正常終了した場合は wchar へのポインタ、それ以外の場合は NULL ポインタを戻します。

戻り値

正常終了した場合は wchar へのポインタ、それ以外の場合は NULL ポインタ。

表 5-17 OCIWideCharStrrchr のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wstr（IN）	検索する wchar 文字列へのポインタ。
wc（IN）	検索する wchar。

OCIWideCharStrCaseConversion

構文

```
size_t OCIWideCharStrCaseConversion(dvoid *hndl, OCIWchar *wdststr, CONST
OCIWchar*wsrcstr, ub4 flag)
```

備考

この関数は、wsrcstr で示されたワイド・キャラクタ文字列をフラグで指定された大文字または小文字に変換し、その結果を wdststr で示された配列にコピーします。結果文字列は、NULL で終了します。

戻り値

結果文字列の文字数（NULL 終了文字を含まない）。

表 5-18 OCIWideCharStrCaseConversion のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wdststr（OUT）	格納先の配列へのポインタ。
wsrcstr（IN）	ソース文字列へのポインタ。

表 5-18 OCIWideCharStrCaseConversion のキーワード / パラメータ

キーワード / パラメータ	意味
flag（IN）	大文字または小文字のどちらに変換するかを指定するフラグ。 ■ OCI_NLS_UPPERCASE: 大文字に変換します。 ■ OCI_NLS_LOWERCASE: 小文字に変換します。 ロケールでの言語設定を大文字 / 小文字の変換に使用するように指定するには、このフラグに OCI_NLS_LINGUISTIC と OR された値を指定します。

OCIWideCharDisplayLength

構文

```
size_t OCIWideCharDisplayLength(dvoid *hndl, OCIWchar wc )
```

備考

この関数は、wc を表示する場合に必要な列位置の数を決定します。列位置の数、または wc が NULL 終了文字の場合は 0 を戻します。

戻り値

表示位置の数。

表 5-19 OCIWideCharDisplayLength のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	wchar 文字。

OCIWideCharMultiByteLength

構文

```
size_t OCIWideCharMultiByteLen(dvoid *hndl, OCIWchar wc )
```

備考

この関数は、wc をマルチバイトでコード化する場合に必要なバイト数を決定します。wc のマルチバイトでのバイト数を戻します。

戻り値

バイト数。

表 5-20 OCIWideCharMultiByteLength のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	wchar 文字。

OCIMultiByteStrcmp

構文

```
int OCIMultiByteStrcmp(dvoid *hndl, CONST text *str1, CONST text *str2, int flag)
```

備考

2 つのマルチバイト文字列を、（コード値に基づいた）バイナリ、言語上の方法、または大文字と小文字を区別しない方法で比較します。

戻り値

- str1 = str2 の場合、0。
- str1 > str2 の場合、正の値。
- str1 < str2 の場合、負の値。

表 5-21 OCIMultiByteStrcmp のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
str1（IN）	NULL で終了する文字列へのポインタ。
str2（IN）	NULL で終了する文字列へのポインタ。
flag（IN）	比較方法を決定するために使用されるフラグ。次のいずれかの値をとることができます。 ■ OCI_NLS_BINARY: バイナリでの比較。これは、デフォルト値です。 ■ OCI_NLS_LANGUISTIC: ロケールで指定された言語上の比較。 大文字と小文字を区別せずに比較するには、このフラグに OCI_NLS_CASE_INSENSITIVE と OR された値を指定します。

OCIMultiByteStrncmp

構文

```
int OCIMultiByteStrncmp(dvoid *hndl, CONST text *str1, size_t len1, text *str2,
size_t len2, int flag)
```

備考

この関数は、str1 の先頭 len1 バイトと str2 の先頭 len2 バイトのみが比較されることを除けば、OCIMultiByteStrcmp() と似ています。NULL 終了文字も比較の対象です。

戻り値

- str1 = str2 の場合、0。
- str1 > str2 の場合、正の値。
- str1 < str2 の場合、負の値。

表 5-22 OCIMultiByeStrncmp のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
str1（IN）	最初の文字列へのポインタ。
len1（IN）	比較する最初の文字列の長さ。
str2（IN）	2 番目の文字列へのポインタ。
len2（IN）	比較する 2 番目の文字列の長さ。
flag（IN）	比較方法を決定するために使用されるフラグ。次のいずれかの値をとることができます。 ■ OCI_NLS_BINARY: バイナリでの比較。これは、デフォルト値です。 ■ OCI_NLS_LANGUISTIC: ロケールで指定された言語上の比較。 大文字と小文字を区別せずに比較するには、このフラグに OCI_NLS_CASE_INSENSITIVE と OR された値を指定します。

OCIMultiByteStrcat

構文

```
size_t OCIMultiByteStrcat(dvoid *hndl, text *dststr, CONST text *srcstr)
```

備考
この関数は、srcstr で示されたマルチバイト文字列のコピーを、dststr で示された文字列の最後に NULL 終了文字も含めて追加します。結果文字列のバイト数（最後の NULL 終了文字を含まない）を戻します。

戻り値
結果文字列のバイト数（最後の NULL 終了文字を含まない)。

表 5-23 OCIMultiByteStrcat のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
dststr（IN/OUT）	追加先のマルチバイト文字列へのポインタ。
srcstr（IN）	追加するソース文字列へのポインタ。

OCIMultiByteStrcpy

構文
```
size_t OCIMultiByteStrcpy(dvoid *hndl, text *dststr, CONST text *srcstr)
```

備考
この関数は、srcstr で示されたマルチバイト文字列を、dststr で示された配列に NULL 終了文字も含めてコピーします。コピーされたバイト数（最後の NULL 終了文字を含まない）を戻します。

戻り値
コピーされたバイト数（最後の NULL 終了文字を含まない)。

表 5-24 OCIMultiByteStrcpy のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドルへのポインタ。
dststr（OUT）	格納先バッファへのポインタ。
srcstr（IN）	ソース・マルチバイト文字列へのポインタ。

OCIMultiByteStrlen

構文
```
size_t OCIMultiByteStrlen(dvoid *hndl, CONST text *str)
```

備考
この関数は、str で示されたマルチバイト文字列のバイト数（NULL 終了文字を含まない）を計算し、その数を戻します。

戻り値
バイト数（最後の NULL 終了文字を含まない）。

表 5-25 OCIMultiByteStrlen のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドルへのポインタ。
str（IN）	ソース・マルチバイト文字列へのポインタ。

OCIMultiByteStrncat

構文
```
size_t OCIMultiByteStrncat(dvoid *hndl, text *dststr, CONST text *srcstr, size_t n)
```

備考
この関数は、最大 n バイトが srcstr から dststr に追加されることを除けば、OCIMultiByteStrcat() に似ています。srcstr に NULL 終了文字があると、そこで追加処理は停止し、n バイト内でできるだけ多くの文字が追加されます。dststr は、NULL で終了します。

戻り値
結果文字列のバイト数（最後の NULL 終了文字を含まない）。

表 5-26 OCIMultiByteStrncat のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドルへのポインタ。
dststr（IN）	追加するソース・マルチバイト文字列へのポインタ。
srcstr（IN/OUT）	追加先のマルチバイト文字列へのポインタ。
n（IN）	srcstr から追加するバイト数。

OCIMultiByteStrncpy

構文
```
size_t OCIMultiByteStrncpy(dvoid *hndl, text *dststr, CONST text *srcstr, size_t n)
```

備考
この関数は、最大 n バイトが srcstr で示された配列から dststr で示された配列にコピーされることを除けば、OCIMultiByteStrcpy() に似ています。srcstr に NULL 終了文字があると、そこでコピー処理は停止し、n バイト内でできるだけ多くの文字がコピーされます。結果文字列は、NULL で終了します。

戻り値
コピーされたバイト数（最後の NULL 終了文字を含まない）。

表 5-27 OCIMultiByteStrncpy のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドルへのポインタ。
dststr（OUT）	宛先バッファへのポインタ。
srcstr（IN）	ソース・マルチバイト文字列へのポインタ。
n（IN）	srcstr からコピーするバイト数。

OCIMultiByteStrnDisplayLength

構文
```
size_t OCIMultiByteStrnDisplayLength(dvoid *hndl, CONST text *str1, size_t n)
```

備考
この関数は、n バイトの範囲内のすべての文字が占有する表示位置の数を戻します。

戻り値
表示位置の数。

表 5-28 OCIMultiByteStrnDisplayLength のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
str（IN）	マルチバイト文字列へのポインタ。
n（IN）	検査するバイト数。

OCIMultiByteStrCaseConversion

構文

```
size_t OCIMultiByteStrCaseConversion(dvoid *hndl, text *dststr, CONST text *srcstr,
ub4 flag)
```

備考

この関数は、srcstr で示されたマルチバイト文字列を flag で指定された大文字または小文字に変換し、その結果を dststr で示された配列にコピーします。結果文字列は、NULL で終了します。

戻り値

結果文字列のバイト数（NULL 終了文字を含まない）。

表 5-29 OCIMultibyteStrCaseConversion のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
dststr（OUT）	格納先の配列へのポインタ。
srcstr（IN）	ソース文字列へのポインタ。
flag（IN）	大文字または小文字のどちらに変換するかを指定するフラグ。 ■ OCI_NLS_UPPERCASE: 大文字に変換します。 ■ OCI_NLS_LOWERCASE: 小文字に変換します。 ロケールでの言語設定を大文字 / 小文字の変換に使用するように指定するには、このフラグに OCI_NLS_LINGUISTIC と OR された値を指定します。

文字列操作のサンプル・コード

次に、文字列を操作する簡単な事例を示します。

```
size_t MyConvertMultiByteToWideChar(envhp, dstBuf, dstSize, srcStr)
OCIEnv     *envhp;
OCIWchar   *dstBuf;
size_t      dstSize;
text       *srcStr;                          /* null terminated source string */
{
  sword  ret;
  size_t dstLen = 0;
  size_t srcLen;

  /* get length of source string */
```

```
  srcLen = OCIMultiByteStrlen(envhp, srcStr);

  ret = OCIMultiByteInSizeToWideChar(envhp,       /* environment handle */
                dstBuf,                               /* destination buffer */
                dstSize,                          /* destination buffer size */
                srcStr,                                   /* source string */
                srcLen,                           /* length of source string */
                &dstLen);                  /* pointer to destination length */

  if (ret != OCI_SUCCESS)
  {
    checkerr(envhp, ret, OCI_HTYPE_ENV);
  }
  return(dstLen);
}
```

詳細は、『Oracle8*i* コール・インタフェース・プログラマーズ・ガイド』および『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

文字の分類

Oracle コール・インタフェースでは、文字を分類するための多くの関数コールが用意されています。

表 5-30 OCI 文字分類コール

関数コール	説明
OCIWideCharIsAlnum()	ワイド・キャラクタが、文字であるか 10 進数字であるかをテストします。
OCIWideCharIsAlpha()	ワイド・キャラクタが、アルファベット文字であるかどうかをテストします。
OCIWideCharIsCntrl()	ワイド・キャラクタが、制御文字であるかどうかをテストします。
OCIWideCharIsDigit()	ワイド・キャラクタが、10 進数字であるかどうかをテストします。
OCIWideCharIsGraph()	ワイド・キャラクタが、図形文字であるかどうかをテストします。
OCIWideCharIsLower()	ワイド・キャラクタが、小文字であるかどうかをテストします。
OCIWideCharIsPrint()	ワイド・キャラクタが、印字可能文字であるかどうかをテストします。
OCIWideCharIsPunct()	ワイド・キャラクタが、句読点文字であるかどうかをテストします。
OCIWideCharIsSpace()	ワイド・キャラクタが、スペース文字であるかどうかをテストします。

表 5-30 OCI 文字分類コール

関数コール	説明
OCIWideCharIsUpper()	ワイド・キャラクタが、大文字であるかどうかをテストします。
OCIWideCharIsXdigit()	ワイド・キャラクタが、16 進数字であるかどうかをテストします。
OCIWideCharIsSingleByte()	wc がマルチバイトに変換されるとき、シングルバイト文字であるかどうかをテストします。

OCIWideCharIsAlnum

構文

```
boolean OCIWideCharIsAlnum(dvoid *hndl, OCIWchar wc)
```

備考

wc が、文字であるか 10 進数字であるかをテストします。

戻り値

TRUE または FALSE。

表 5-31 OCIWideCharIsAlnum のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsAlpha

構文

```
boolean OCIWideCharIsAlpha(dvoid *hndl, OCIWchar wc)
```

備考

wc が、アルファベット文字であるかどうかをテストします。

戻り値

TRUE または FALSE。

表 5-32 OCIWideCharIsAlpha のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsCntrl

構文

```
boolean OCIWideCharIsCntrl(dvoid *hndl, OCIWchar wc)
```

備考

wc が、制御文字であるかどうかをテストします。

戻り値

TRUE または FALSE。

表 5-33 OCIWideCharIsCntrl のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsDigit

構文

```
boolean OCIWideCharIsDigit(dvoid *hndl, OCIWchar wc)
```

備考

wc が、10 進数字であるかどうかをテストします。

戻り値

TRUE または FALSE。

表 5-34 OCIWideCharIsDigit のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。

表 5-34 OCIWideCharIsDigit のキーワード / パラメータ（続き）

キーワード / パラメータ	意味
wc（IN）	テストする wchar。

OCIWideCharIsGraph

構文

```
boolean OCIWideCharIsGraph(dvoid *hndl, OCIWchar wc)
```

備考

wc が、図形文字であるかどうかをテストします。図形文字は、視覚表現を持つ文字で、通常、アルファベット文字、10 進数字および句読点がこれに含まれます。

戻り値

TRUE または FALSE。

表 5-35 OCIWideCharIsGraph のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsLower

構文

```
boolean OCIWideCharIsLower(dvoid *hndl, OCIWchar wc)
```

備考

wc が、小文字であるかどうかをテストします。

戻り値

TRUE または FALSE。

表 5-36 OCIWideCharIsLower のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsPrint

構文

```
boolean OCIWideCharIsPrint(dvoid *hndl, OCIWchar wc)
```

備考

wc が、印刷可能文字であるかどうかをテストします。

戻り値

TRUE または FALSE。

表 5-37 OCIWideCharIsPrint のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsPunct

構文

```
boolean OCIWideCharIsPunct(dvoid *hndl, OCIWchar wc)
```

備考

wc が、句読点文字であるかどうかをテストします。

戻り値

TRUE または FALSE。

表 5-38 OCIWideCharIsPunct のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsSpace

構文

```
boolean OCIWideCharIsSpace(dvoid *hndl, OCIWchar wc)
```

備考

wc が、スペース文字であるかどうかをテストします。スペース文字は、テキスト中に余白（空白、タブ、改行、垂直タブ、改ページなど）のみを表示します。

戻り値

TRUE または FALSE。

表 5-39 OCIWideCharIsSpace のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsUpper

構文

```
boolean OCIWideCharIsUpper(dvoid *hndl, OCIWchar wc)
```

備考

wc が、大文字であるかどうかをテストします。

戻り値

TRUE または FALSE。

表 5-40 OCIWideCharIsUpper のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsXdigit

構文

```
boolean OCIWideCharIsXdigit(dvoid *hndl, OCIWchar wc)
```

備考

wc が 16 進数字（0 ～ 9、A ～ F、a ～ f）であるかどうかをテストします。

戻り値

TRUE または FALSE。

表 5-41 OCIWideCharIsXdigit のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

OCIWideCharIsSingleByte

構文

```
boolean OCIWideCharIsSingleByte(dvoid *hndl, OCIWchar wc)
```

備考

wc がマルチバイトに変換されるとき、シングルバイト文字であるかどうかをテストします。

戻り値

TRUE または FALSE。

表 5-42 OCIWideCharIsSingleByte のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	キャラクタ・セットを決定するための OCI 環境ハンドルまたはユーザー・セッション・ハンドル。
wc（IN）	テストする wchar。

文字分類のサンプル・コード

```
 /* Character classification sample code */
boolean MyIsNumberWideCharString(envhp, srcStr)
OCIEnv   *envhp;
OCIWchar *srcStr;                                   /* wide char source string */
{
```

```
  OCIWchar *pstr = srcStr;                          /* define and init pointer */
  boolean status = TRUE;                    /* define and init status variable */

  /* Check input */
  if (pstr == (OCIWchar*) NULL)
    return(FALSE);


  if (*pstr == (OCIWchar) NULL)
    return(FALSE);

                                              /* check each character for digit */
  do
  {
    if (OCIWideCharIsDigit(envhp, *pstr) != TRUE)
    {
      status = FALSE;
      break;                                    /* non decimal digit character */
    }
  } while ( *++pstr != (OCIWchar) NULL);

  return(status);
}
```

詳細は、『Oracle8*i* コール・インタフェース・プログラマーズ・ガイド』および『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

キャラクタ・セット変換

Oracle キャラクタ・セットと Unicode（16 ビットの固定幅 Unicode）間の変換がサポートされています。Unicode から Oracle キャラクタ・セットへのマッピングができない場合は、置換文字が使用されます。したがって、ラウンドトリップ変換が常に可能なわけではありません。

表 5-43 OCI キャラクタ・セット変換コール

関数コール	説明
OCICharsetToUnicode()	src で示されたマルチバイト文字列を Unicode に変換し、dst で示された配列に格納します。
OCIUnicodeToCharset()	src で示された Unicode 文字列をマルチバイトに変換し、dst で示された配列に格納します。
OCICharSetConversionIs ReplacementUsed()	前回の OCICharsetConv() の起動の際に、キャラクタ・セット変換で変換できなかった文字に対して置換文字が使用されたかどうかを示します。

OCICharSetToUnicode

構文

```
sword OCICharSetToUnicode(dvoid *hndl, ub2 *dst, size_t dstlen, CONST text *src,
size_t srclen, size_t *rsize)
```

備考

この関数は、src で示されたマルチバイト文字列を Unicode に変換し、dst で示された配列に格納します。ソース制限または宛先制限に到達すると、変換は停止します。関数は、Unicode に変換された文字数を戻します。dstlen がゼロの場合、変換結果の文字数が rsize に戻されますが、実際の変換は行われません。

戻り値

OCI_SUCCESS、OCI_INVALID_HANDLE または OCI_ERROR。

表 5-44 OCICharSetToUnicode のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドルへのポインタ。
dst（OUT）	宛先バッファへのポインタ。
dstlen（IN）	宛先バッファ・サイズ（文字単位）。
src（IN）	ソース・マルチバイト文字列へのポインタ。
srclen（IN）	ソース文字列サイズ（バイト単位）。
rsize（OUT）	変換された文字数。NULL ポインタの場合は、戻り値がないことを意味します。

OCIUnicodeToCharSet

構文

```
sword OCIUnicodeToCharSet(dvoid *hndl, text *dst, size_t dstlen, CONST ub2 *src,
size_t srclen, size_t *rsize)
```

備考

この関数は、src で示された Unicode 文字列をマルチバイトに変換し、dst で示された配列に格納します。ソース制限または宛先制限に到達すると、変換は停止します。関数は、マルチバイトに変換されたバイト数を戻します。dstlen がゼロの場合、変換結果のバイト数が rsize に戻されますが、実際の変換は行われません。

Unicode 文字が OCI 環境ハンドルまたはユーザー・セッション・ハンドルで指定されたキャラクタ・セットで変換できない場合は、置換文字が使用されます。この場合、OCICharsetConversionIsReplacementUsed() は TRUE を戻します。

戻り値

OCI_SUCCESS、OCI_INVALID_HANDLE または OCI_ERROR。

表 5-45 OCIUnicodeToCharSet のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドルへのポインタ。
dst（OUT）	宛先バッファへのポインタ。
dstlen（IN）	宛先バッファ・サイズ（バイト単位）。
src（IN）	Unicode 文字列へのポインタ。
srclen（IN）	ソース文字列サイズ（文字単位）。
rsize（OUT）	変換されたバイト数。NULL ポインタの場合は、戻り値がないことを意味します。

OCICharSetConversionIsReplacementUsed

構文

```
boolean OCICharSetConversionIsReplacementUsed(dvoid *hndl)
```

備考

この関数は、前回の OCICharSetToUnicode() の起動の際に、キャラクタ・セット変換で変換できなかった文字に対して置換文字が使用されたかどうかを示します。

戻り値

前回の OCICharsetConv() の起動で置換文字が使用された場合は TRUE、使用されなかった場合は FALSE。

表 5-46 OCICharSetConversionIsReplacementUsed のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	OCI 環境ハンドルまたはユーザー・セッション・ハンドルへのポインタ。

Oracle キャラクタ・セットと Unicode（16 ビットの固定幅 Unicode）間の変換がサポートされています。Unicode から Oracle キャラクタ・セットへのマッピングができない場合は、置換文字が使用されます。したがって、ラウンドトリップ変換が常に可能なわけではありません。

キャラクタ・セット変換のサンプル・コード

次に、Unicode への変換の簡単な事例を示します。

```
size_t MyConvertMultiByteToUnicode(envhp, dstBuf, dstSize, srcStr)
OCIEnv *envhp;
ub2    *dstBuf;
size_t  dstSize;
text   *srcStr;
{
  sword  ret;
  size_t dstLen = 0;
  size_t srcLen;

  /* get length of source string */
  srcLen = OCIMultiByteStrlen(envhp, srcStr);


  ret = OCICharSetToUnicode(envhp,                    /* environment handle */
                dstBuf,                               /* destination buffer */
                dstSize,                        /* size of destination buffer */
                srcStr,                                  /* source string */
                srcLen,                           /* length of source string */
                &dstLen);                  /* pointer to destination length */

  if (ret != OCI_SUCCESS)
  {
    checkerr(envhp, ret, OCI_HTYPE_ENV);
  }
  return(dstLen);
}
```

詳細は、『Oracle8*i* コール・インタフェース・プログラマーズ・ガイド』および『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

メッセージ・メカニズム

ユーザー・メッセージ API では、カートリッジ開発者のために簡易インタフェースが提供されています。これによって、Oracle メッセージと同様にユーザー自身のメッセージを取出すことができます。

表 5-47 OCI メッセージ関数コール

関数コール	説明
OCIMessageOpen()	hndl で示された言語で、製品の機能に対するメッセージ・ハンドルをオープンします。
OCIMessageGet()	msgno で識別するメッセージ番号の付いたメッセージを取り出します。バッファがゼロでない場合、この関数は msgbuf で示されたバッファにメッセージをコピーします。
OCIMessageClose()	msgh で示されたメッセージ・ハンドルをクローズし、そのハンドルに対応付けられているメモリーをすべて解放します。

詳細は、『Oracle8*i* コール・インタフェース・プログラマーズ・ガイド』および『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

OCIMessageOpen

構文

```
sword OCIMessageOpen(dvoid *hndl, OCIError *errhp, OCIMsg **msghp, CONST text
*product, CONST text *facility, OCIDuration dur)
```

備考

この関数は、hndl で示された言語で、製品の機能に対するメッセージ・ハンドルをオープンします。まず、その機能に対して hndl に対応するメッセージ・ファイルをオープンします。メッセージ・ファイルのオープンが成功すると、そのファイルを使用してメッセージ・ハンドルを初期化します。メッセージ・ファイルのオープンに失敗した場合は、その機能に対する米語用のメッセージ・ファイルがデフォルトのメッセージ・ファイルとして使用されます。この関数は、メッセージ・ハンドルへのポインタを msghp パラメータに戻します。

戻り値

OCI_SUCCESS、OCI_INVALID_HANDLE または OCI_ERROR。

表 5-48 OCIMessageOpen のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	メッセージ言語用の OCI 環境ハンドルまたはユーザー・セッション・ハンドルへのポインタ。
errhp（IN/OUT）	OCI エラー・ハンドル。エラーがある場合は、errhp に記録され、NULL ポインタが戻されます。診断情報は OCIErrorGet() をコールすると取得できます。
msghp（OUT）	戻されるメッセージ・ハンドル。

表 5-48 OCIMessageOpen のキーワード / パラメータ

キーワード / パラメータ	意味
product（IN）	製品名へのポインタ。製品名は、システムによって異なる方法でメッセージ用のディレクトリの位置を見つけるために使用されます。たとえば Solaris の場合、製品「rdbms」のメッセージ・ファイルのディレクトリは「$ORACLE_HOME/rdbms」です。
facility（IN）	製品の機能名へのポインタ。これは、メッセージ・ファイル名を組み立てるために使用されます。メッセージ・ファイル名では、接頭辞として機能名が使用されます。たとえば、機能「img」の米語でのメッセージ・ファイル名は、「imgus.msb」になります。この場合、「us」は米語の略称で、「msb」はメッセージのバイナリ・ファイルという意味の拡張子です。
dur（IN）	戻されたメッセージ・ハンドルのメモリー割当て期間。次のいずれかの値をとることができます。 ■ OCI_DURATION_PROCESS ■ OCI_DURATION_STATEMENT ■ OCI_DURATION_SESSION

OCIMessageGet

構文

```
text *OCIMessageGet(OCIMsg *msgh, ub4 msgno, text *msgbuf, size_t buflen)
```

備考

この関数は、msgno で識別されるメッセージ番号の付いたメッセージを取得します。buflen がゼロでない場合、msgbuf で示されたバッファにメッセージをコピーします。buflen がゼロの場合、取得したメッセージは msgh で示されたメッセージ・ハンドル内のメッセージ・バッファにコピーされます。どちらの場合も、この関数は NULL で終了するメッセージ文字列へのポインタを戻します。必要なメッセージが取得できなかった場合は、NULL ポインタを戻します。

戻り値

成功の場合は NULL で終了するメッセージ文字列へのポインタ、それ以外の場合は NULL ポインタ。

表 5-49 OCIMessageGet のキーワード / パラメータ

キーワード / パラメータ	意味
msgh（IN/OUT）	OCIMessageOpen() によってあらかじめオープンされているメッセージ・ハンドルへのポインタ。
msgno（IN）	メッセージを取得するためのメッセージ番号。
msgbuf（OUT）	取り出したメッセージを格納する宛先バッファへのポインタ。buflen がゼロの場合、NULL ポインタになります。
buflen（IN）	前述の宛先バッファのサイズ。

OCIMessageClose

構文

```
sword OCIMessageClose(dvoid *hndl, OCIError *errhp, OCIMsg *msgh)
```

備考

この関数は、msgh で示されたメッセージ・ハンドルをクローズし、そのハンドルに対応付けられているメモリーをすべて解放します。

戻り値

OCI_SUCCESS、OCI_INVALID_HANDLE または OCI_ERROR。

表 5-50 OCIMessageClose のキーワード / パラメータ

キーワード / パラメータ	意味
hndl（IN/OUT）	メッセージ言語用の OCI 環境ハンドルまたはユーザー・セッション・ハンドルへのポインタ。
errhp（IN/OUT）	OCI エラー・ハンドル。エラーがある場合は、errhp に記録され、NULL ポインタが戻されます。診断情報は OCIErrorGet() をコールすると取得できます。
msgh（IN/OUT）	OCIMessageOpen() によってあらかじめオープンされているメッセージ・ハンドルへのポインタ。

LMSGEN

備考

lmsgen ユーティリティは、テキスト・ベースのメッセージ・ファイル（.msg）をバイナリ形式（.msb）に変換します。

構文

```
LMSGEN <text file> <product> <facility> [language]
```

ここでは、次のように指定します

<text file> はメッセージ・テキスト・ファイルを指定します。
<product> は製品名を指定します。
<facility> は機能名を指定します。
[language] は書式 <language>_<territory>.<character set> を使用して任意のメッセージ言語を指定します。

この処理は、メッセージ・ファイルが言語で正しくタグ付けされていない場合に必要です。

テキスト・メッセージ・ファイルの形式

- '/' および '/ /' で始まる行は内部コメントとして扱われるため、その行は無視されます。
- 特定の言語でメッセージ・ファイルにタグを付けるには、次のように指定します。

  ```
  #   CHARACTER_SET_NAME= Japanese_Japan.JA16EUC
  ```

- それぞれのメッセージは、次の 3 つのフィールドから構成されます。

  ```
  <message #>, <warning level #>, <message text>
  ```

  - message # は、メッセージ・ファイル内で一意な値である必要があります。
  - warning level # は、現在使用されていないので、単に 0 を使用します。
  - message text は、76 バイトを超えてはなりません。

例

```
/ Copyright (c) 1988 by the Oracle Corporation.  All rights reserved.
/ This is a testing us7ascii message file
# CHARACTER_SET_NAME= american_america.us7ascii
/
00000, 00000, "Export terminated unsuccessfully\n"
00003, 00000, "no storage definition found for segment(%lu, %lu)"
```

メッセージの例

設定

ここでは、.msb メッセージ・ファイルからメッセージを取り出す例を示します。次の設定を使用します。

```
product = $HOME/myApp
facility = imp
Language = American language
```

前述の設定に基づいて、メッセージ・ファイル **$HOME/myApp/mesg/impus.msb** が使用されます。

メッセージ・ファイル

lmsgen は、メッセージ・ファイル（impus.msg）をバイナリ形式（impus.msb）に変換します。

次に、テキスト・メッセージ・ファイル impus.msg の一部分を示します。

```
...
00128,2, "Duplicate entry %s found in %s"
...
```

メッセージのサンプル・コード

```
/* Assume that the OCI environment or user session handle, product, facility and
cache size are all initialized properly. */
...
OCIMsg msghnd;                                                  /* message handle */
         /* initialize a message handle for retrieving messages from impus.msg*/
err = OCIMessageOpen(hndl,errhp, &msghnd, prod,fac,OCI_DURATION_SESSION);
if (err != OCI_SUCCESS)
                                                                /* error handling */
...
                             /* retrieve the message with message number = 128 */
msgptr = OCIMessageGet(msghnd, 128, msgbuf, sizeof(msgbuf));
                           /* do something with the message, such as display it */
...
          /* close the message handle when we has no more message to retrieve */
OCIMessageClose(hndl, errhp, msghnd);
```

A

ロケール・データ

この付録では、Oracle Server でサポートされている言語、地域、キャラクタ・セットおよびその他のロケール・データについて説明します。トピックは、次のとおりです。

- 言語
- 翻訳済みメッセージ
- 地域
- キャラクタ・セット
- 言語の定義
- 暦法

サポートされているキャラクタ・セット、言語、地域およびソート順序についての情報は、動的データ・ビュー V$NLS_VALID_VALUES を問い合せることによって取得できます。このビューによって戻される情報の詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。

言語

表 A-1 に、Oracle Server がサポートしている言語を示します。

表 A-1 Oracle がサポートしている言語

名前	略称
AMERICAN	us
ARABIC	ar
BENGALI	bn
BRAZILIAN PORTUGUESE	ptb
BULGARIAN	bg
CANADIAN FRENCH	frc
CATALAN	ca
CROATIAN	hr
CZECH	cs
DANISH	dk
DUTCH	nl
EGYPTIAN	eg
ENGLISH	gb
ESTONIAN	et
FINNISH	sf
FRENCH	f
GERMAN DIN	din
GERMAN	d
GREEK	el
HEBREW	iw
HUNGARIAN	hu
ICELANDIC	is
INDONESIAN	in
ITALIAN	i
JAPANESE	ja

表 A-1 Oracle がサポートしている言語

名前	略称
KOREAN	ko
LATIN AMERICAN SPANISH	esa
LATVIAN	lv
LITHUANIAN	lt
MALAY	ms
MEXICAN SPANISH	esm
NORWEGIAN	n
POLISH	pl
PORTUGUESE	pt
ROMANIAN	ro
RUSSIAN	ru
SIMPLIFIED CHINESE	zhs
SLOVAK	sk
SLOVENIAN	sl
SPANISH	e
SWEDISH	s
THAI	th
TRADITIONAL CHINESE	zht
TURKISH	tr
UKRAINIAN	uk
VIETNAMESE	vn

翻訳済みメッセージ

oracle エラー・メッセージおよびユーザー・インタフェースは、表 A-2 に示された言語で翻訳されています。

表 A-2 Oracle がサポートしているメッセージ

名前	略称
ARABIC	ar
BRAZILIAN PORTUGUESE	ptb
CATALAN	ca
CZECH	cs
DANISH	dk
DUTCH	nl
FINNISH	sf
FRENCH	f
GERMAN	d
GREEK	el
HEBREW	iw
HUNGARIAN	hu
ITALIAN	i
JAPANESE	ja
KOREAN	ko
LATIN AMERICAN SPANISH	esa
NORWEGIAN	n
POLISH	pl
PORTUGUESE	pt
ROMANIAN	ro
RUSSIAN	ru
SIMPLIFIED CHINESE	zhs
SLOVAK	sk
SPANISH	e
SWEDISH	s

表 A-2 Oracle がサポートしているメッセージ

名前	略称
TRADITIONAL CHINESE	zht
TURKISH	tr

地域

表 A-3 に、Oracle Server がサポートしている地域を示します。

表 A-3 Oracle がサポートしている地域

名前		
ALGERIA	HUNGARY	QATAR
AMERICA	ICELAND	ROMANIA
AUSTRALIA	INDONESIA	SAUDI ARABIA
AUSTRIA	IRAQ	SINGAPORE
BAHRAIN	IRELAND	SLOVAKIA
BANGLADESH	ISRAEL	SLOVENIA
BELGIUM	ITALY	SOMALIA
BRAZIL	JAPAN	SOUTH AFRICA
BULGARIA	JORDAN	SPAIN
CANADA	KAZAKHSTAN	SUDAN
CATALONIA	KUWAIT	SWEDEN
CHINA	LATVIA	SWITZERLAND
CIS	LEBANON	SYRIA
CROATIA	LIBYA	TAIWAN
CYPRUS	KOREA	THAILAND
CZECH	LITHUANIA	THE NETHERLANDS
CZECHOSLOVAKIA	LUXEMBOURG	TUNISIA
DENMARK	MALAYSIA	TURKEY
DJIBOUTI	MAURITANIA	UKRAINE
EGYPT	MEXICO	UNITED ARAB EMIRATES

表 A-3 Oracle がサポートしている地域

名前		
ESTONIA	MOROCCO	UNITED KINGDOM
FINLAND	NEW ZEALAND	UZBEKISTAN
FRANCE	NORWAY	VIETNAM
GERMANY	OMAN	YEMEN
GREECE	POLAND	
HONG KONG	PORTUGAL	

キャラクタ・セット

後述の Oracle がサポートしているキャラクタ・セットは、簡単に参照できるように、次の広義の 3 つの言語グループに従って示されています。

- アジア地域言語のキャラクタ・セット
- ヨーロッパ地域言語のキャラクタ・セット
- 中東地域言語のキャラクタ・セット

キャラクタ・セットによっては複数の言語に対して示されているものがあります。これは、そのキャラクタ・セットが多言語をサポートしているためです。たとえば、Unicode は世界の主なスクリプトのほとんどをサポートしているので、アジア地域、ヨーロッパ地域および中東地域の言語グループにわたって記述されています。

コメントには、使用されるタイプのコード化方式が次のように示されています。

SB = シングルバイト・コード化方式

MB = マルチバイト・コード化方式

FIXED = 固定幅マルチバイト・コード化方式

第 3 章「キャラクタ・セットの選択」で説明したように、コード化方式のタイプはパフォーマンスに影響を及ぼすので、最も効果的なコード化を使用してご使用の言語のニーズを満たすようにしてください。また、一部のコード化方式は、特定のデータ型のみで使用できます。たとえば、固定幅マルチバイトでのコード化キャラクタ・セットは NCHAR キャラクタ・セットとして使用できますが、データベース・キャラクタ・セットとしては使用できません。

コメントには、そのキャラクタ・セットに固有の他の機能も記述されています。それらは、ユーザーまたはデータベース管理者にとって重要な場合があります。たとえば、そのキャラクタ・セットが新しいユーロ通貨記号をサポートしているかどうか、ユーザー定義文字がキャラクタ・セットのカスタマイズでサポートされているかどうか、およびキャラクタ・セットが（移行の場合に ALTER DATABASE [NATIONAL] CHARACTER SET コマンドを使

用できる）ASCII の完全なスーパーセットであるかどうかが示されています。

EURO = ユーロ記号をサポート

UDC = ユーザー定義文字をサポート

ASCII = ASCII の完全なスーパーセット

個々のコード・ページのレイアウトは記述されていません。特定のキャラクタ・セットの固有情報、文字レパートリおよびコード・ポイントの値については、実際の各国、国際またはベンダー固有の規格を参照してください。

アジア地域言語のキャラクタ・セット

表 A-4 に、アジア地域の言語をサポートしている Oracle キャラクタ・セットを示します。

表 A-4 アジア地域言語のキャラクタ・セット

名前	説明	コメント
BN8BSCII	バングラデシュ国内規格コード 8 ビット BSCII	SB、ASCII
ZHT16BIG5	BIG5 16 ビット中国語（繁体字）	MB、ASCII
ZHS16CGB231280	CGB2312-80 16 ビット中国語（簡体字）	MB、ASCII
JA16EUC	EUC 24 ビット日本語	MB、ASCII
JA16EUCYEN	EUC 24 ビット日本語。'\' は日本語の円記号になります。	MB
JA16EUCFIXED	EUC 16 ビット日本語。JA16EUC の固定幅サブセット（JA16EUC の 2 バイト文字のみが含まれる）。7 ビットまたは 8 ビットの ASCII 文字は含まれません。	FIXED
ZHT32EUC	EUC 32 ビット中国語（繁体字）	MB、ASCII
ZHS16GBK	GBK 16 ビット中国語（簡体字）	MB、ASCII、UDC
ZHS16GBKFIXED	GBK 16 ビット中国語（簡体字）。16 ビット固定幅でシングルバイトではありません。	FIXED、UDC
ZHT16CCDC	HP CCDC 16 ビット中国語（繁体字）	MB、ASCII
JA16DBCS	IBM EBCDIC 16 ビット日本語	MB、UDC
JA16EBCDIC930	IBM DBCS コード・ページ 290 16 ビット日本語	MB、UDC
JA16DBCSFIXED	IBM EBCDIC 16 ビット日本語。16 ビット固定幅でシングルバイトではありません。	FIXED、UDC
KO16DBCS	IBM EBCDIC 16 ビット韓国語	MB、UDC

表 A-4 アジア地域言語のキャラクタ・セット

名前	説明	コメント
KO16DBCSFIXED	IBM EBCDIC 16 ビット韓国語。16 ビット固定幅でシングルバイトではありません。	FIXED、UDC
ZHS16DBCS	IBM EBCDIC 16 ビット中国語（簡体字）	MB、UDC
ZHS16DBCSFIXED	IBM EBCDIC 16 ビット中国語（簡体字）。16 ビット固定幅でシングルバイトではありません。	FIXED、UDC
ZHT16DBCS	IBM EBCDIC 16 ビット中国語（繁体字）	MB、UDC
KO16KSC5601	KSC5601 16 ビット韓国語	MB、ASCII
KO16KSCCS	KSCCS 16 ビット韓国語	MB、ASCII
JA16VMS	JVMS 16 ビット日本語	MB、ASCII
ZHS16MACCGB231280	Mac クライアント CGB2312-80 16 ビット中国語（簡体字）	MB
JA16MACSJIS	Mac クライアント・シフト JIS 16 ビット日本語	MB
TH8MACTHAI	Mac クライアント 8 ビット・ラテン語 / タイ語	SB
TH8MACTHAIS	Mac サーバー 8 ビット・ラテン語 / タイ語	SB、ASCII
ZHT16MSWIN950	MS Windows コード・ページ 950 中国語（繁体字）	MB、ASCII、UDC
KO16MSWIN949	MS Windows コード・ページ 949 韓国語	MB、ASCII、UDC
VN8MSWIN1258	MS Windows コード・ページ 1253 8 ビット・ベトナム語	SB、ASCII、EURO
IN8ISCII	複数スクリプト・インド標準 8 ビット・ラテン / インド諸言語	SB、ASCII
JA16SJIS	シフト JIS 16 ビット日本語	MB、ASCII、UDC
JA16SJISFIXED	シフト JIS 16 ビット日本語。JA16JIS の固定幅サブセット (JA16JIS の 2 バイト文字のみが含まれる)。7 ビットまたは 8 ビットの ASCII 文字は含まれません。	FIXED、UDC
JA16SJISYEN	シフト JIS 16 ビット日本語。'\' は日本語の円記号になります。	MB、UDC
ZHT32SOPS	SOPS 32 ビット中国語（繁体字）	MB、ASCII
ZHT16DBT	台湾課税 16 ビット中国語（繁体字）	MB、ASCII
TH8TISASCII	タイ工業規格 620-2533 - ASCII 8 ビット	SB、ASCII、EURO
TH8TISEBCDIC	タイ工業規格 620-2533 - EBCDIC 8 ビット	SB
ZHT32TRIS	TRIS 32 ビット中国語（繁体字）	MB、ASCII
ZHT32TRISFIXED	TRIS 32 ビット固定幅中国語（繁体字）	FIXED

表 A-4 アジア地域言語のキャラクタ・セット

名前	説明	コメント
AL24UTFFSS	Unicode 1.1 UTF-8 ユニバーサル・キャラクタ・セット	MB、ASCII、EURO
UTF8	Unicode 2.0 UTF-8 ユニバーサル・キャラクタ・セット	MB、ASCII、EURO
VN8VN3	VN3 8 ビット・ベトナム語	SB、ASCII

ヨーロッパ地域言語のキャラクタ・セット

表 A-5 に、ヨーロッパ地域の言語をサポートしている Oracle キャラクタ・セットを示します。

表 A-5 ヨーロッパ地域言語のキャラクタ・セット

名前	説明	コメント
US7ASCII	ASCII 7 ビット米語	SB、ASCII
SF7ASCII	ASCII 7 ビット・フィンランド語	SB
YUG7ASCII	ASCII 7 ビット・ユーゴスラビア語	SB
RU8BESTA	BESTA 8 ビット・ラテン語 / キリル文字	SB、ASCII
EL8GCOS7	Bull EBCDIC GCOS7 8 ビット・ギリシャ語	SB
WE8GCOS7	Bull EBCDIC GCOS7 8 ビット西欧	SB
EL8DEC	DEC 8 ビット・ラテン語 / ギリシャ語	SB
TR7DEC	DEC VT100 7 ビット・トルコ語	SB
TR8DEC	DEC 8 ビット・トルコ語	SB、ASCII
TR8EBCDIC	EBCDIC コード・ページ 1026 8 ビット・トルコ語	SB
TR8PC857	IBM-PC コード・ページ 857 8 ビット・トルコ語	SB、ASCII
TR8MACTURKISH	Mac クライアント 8 ビット・トルコ語	SB
TR8MACTURKISHS	Mac サーバー 8 ビット・トルコ語	SB、ASCII
TR8MSWIN1254	MS Windows コード・ページ 1254 8 ビット・トルコ語	SB、ASCII、EURO
WE8BS2000L5	Siemens EBCDIC.DF.L5 8 ビット西欧 / トルコ語	SB
WE8DEC	DEC 8 ビット西欧	SB、ASCII
D7DEC	DEC VT100 7 ビット・ドイツ語	SB
F7DEC	DEC VT100 7 ビット・フランス語	SB
S7DEC	DEC VT100 7 ビット・スウェーデン語	SB

表 A-5 ヨーロッパ地域言語のキャラクタ・セット

名前	説明	コメント
E7DEC	DEC VT100 7 ビット・スペイン語	SB
NDK7DEC	DEC VT100 7 ビット・ノルウェー語 / デンマーク語	SB
I7DEC	DEC VT100 7 ビット・イタリア語	SB
NL7DEC	DEC VT100 7 ビット・オランダ語	SB
CH7DEC	DEC VT100 7 ビット・スイス語（ドイツ語 / フランス語）	SB
SF7DEC	DEC VT100 7 ビット・フィンランド語	SB
WE8DG	DG 8 ビット西欧	SB、ASCII
WE8EBCDIC37C	EBCDIC コード・ページ 37 8 ビット Oracle / c	SB
WE8EBCDIC37	EBCDIC コード・ページ 37 8 ビット西欧	SB
D8EBCDIC273	EBCDIC コード・ページ 273 / 1 8 ビット・オーストリア・ドイツ語	SB
DK8EBCDIC277	EBCDIC コード・ページ 277 / 1 8 ビット・デンマーク語	SB
S8EBCDIC278	EBCDIC コード・ページ 278 / 1 8 ビット・スウェーデン語	SB
I8EBCDIC280	EBCDIC コード・ページ 280 / 1 8 ビット・イタリア語	SB
WE8EBCDIC284	EBCDIC コード・ページ 284 8 ビット・ラテンアメリカ / スペイン語	SB
WE8EBCDIC285	EBCDIC コード・ページ 285 8 ビット西欧	SB
F8EBCDIC297	EBCDIC コード・ページ 297 8 ビット・フランス語	SB
WE8EBCDIC500C	EBCDIC コード・ページ 500 8 ビット Oracle / c	SB
WE8EBCDIC500	EBCDIC コード・ページ 500 8 ビット西欧	SB
EE8EBCDIC870	EBCDIC コード・ページ 870 8 ビット東欧	SB
WE8EBCDIC871	EBCDIC コード・ページ 871 8 ビット・アイスランド語	SB
EL8EBCDIC875	EBCDIC コード・ページページ 875 8 ビット・ギリシャ語	SB
CL8EBCDIC1025	EBCDIC コード・ページ 1025 8 ビット・キリル文字	SB
CL8EBCDIC1025X	EBCDIC コード・ページ 1025（修正版）8 ビット・キリル文字	SB
BLT8EBCDIC1112	EBCDIC コード・ページ 1112 8 ビット・バルト多言語	SB
D8EBCDIC1141	EBCDIC コード・ページ 1141 8 ビット・オーストリア・ドイツ語	SB、EURO
DK8EBCDIC1142	EBCDIC コード・ページ 1142 8 ビット・デンマーク語	SB、EURO

表 A-5 ヨーロッパ地域言語のキャラクタ・セット

名前	説明	コメント
S8EBCDIC1143	EBCDIC コード・ページ 1143 8 ビット・スウェーデン語	SB、EURO
I8EBCDIC1144	EBCDIC コード・ページ 1144 8 ビット・イタリア語	SB、EURO
F8EBCDIC1147	EBCDIC コード・ページ 1147 8 ビット・フランス語	SB、EURO
EEC8EUROASCI	EEC Targon 35 ASCI 西欧 / ギリシャ語	SB
EEC8EUROPA3	EEC EUROPA3 8 ビット西欧 / ギリシャ語	SB
LA8PASSPORT	ドイツ政府プリンタ 8 ビット全欧州ラテン語	SB、ASCII
WE8HP	HP LaserJet 8 ビット西欧	SB
WE8ROMAN8	HP Roman8 8 ビット西欧	SB、ASCII
HU8CWI2	ハンガリー語 8 ビット CWI-2	SB、ASCII
HU8ABMOD	ハンガリー語 8 ビット特別 AB Mod	SB、ASCII
LV8RST104090	IBM-PC 代替コード・ページ 8 ビット・ラトビア語（ラテン語 / キリル文字）	SB、ASCII
US8PC437	IBM-PC コード・ページ 437 8 ビット米語	SB、ASCII
BG8PC437S	IBM-PC コード・ページ 437 8 ビット（ブルガリア語の修正版）	SB、ASCII
EL8PC437S	IBM-PC コード・ページ 437 8 ビット（ギリシャ語の修正版）	SB、ASCII
EL8PC737	IBM-PC コード・ページ 737 8 ビット・ギリシャ語 / イタリア語	SB
LT8PC772	IBM-PC コード・ページ 772 8 ビット・リトアニア語（ラテン語 / キリル文字）	SB、ASCII
LT8PC774	IBM-PC コード・ページ 774 8 ビット・リトアニア語（ラテン語）	SB、ASCII
BLT8PC775	IBM-PC コード・ページ 775 8 ビット・バルト語	SB、ASCII
WE8PC850	IBM-PC コード・ページ 850 8 ビット西欧	SB、ASCII
EL8PC851	IBM-PC コード・ページ 851 8 ビット・ギリシャ語 / ラテン語	SB、ASCII
EE8PC852	IBM-PC コード・ページ 852 8 ビット東欧	SB、ASCII
RU8PC855	IBM-PC コード・ページ 855 8 ビット・ラテン語 / キリル文字	SB、ASCII
WE8PC858	IBM-PC コード・ページ 858 8 ビット西欧	SB、ASCII、EURO
WE8PC860	IBM-PC コード・ページ 860 8 ビット西欧	SB、ASCII
IS8PC861	IBM-PC コード・ページ 861 8 ビット・アイスランド語	SB、ASCII
CDN8PC863	IBM-PC コード・ページ 863 8 ビット・カナダ・フランス語	SB、ASCII

表 A-5 ヨーロッパ地域言語のキャラクタ・セット

名前	説明	コメント
N8PC865	IBM-PC コード・ページ 865 8 ビット・ノルウェー語	SB、ASCII
RU8PC866	IBM-PC コード・ページ 866 8 ビット・ラテン語 / キリル文字	SB、ASCII
EL8PC869	IBM-PC コード・ページ 869 8 ビット・ギリシャ語 / ラテン語	SB、ASCII
LV8PC1117	IBM-PC コード・ページ 1117 8 ビット・ラトビア語	SB、ASCII
US8ICL	ICL EBCDIC 8 ビット米語	SB
WE8ICL	ICL EBCDIC 8 ビット西欧	SB
WE8ISOICLUK	ICL 特別バージョン ISO8859-1	SB
WE8ISO8859P1	ISO 8859-1 西欧	SB、ASCII
EE8ISO8859P2	ISO 8859-2 東欧	SB、ASCII
SE8ISO8859P3	ISO 8859-3 南欧	SB、ASCII
NEE8ISO8859P4	ISO 8859-4 北欧および北東欧州	SB、ASCII
CL8ISO8859P5	ISO 8859-5 ラテン語 / キリル文字	SB、ASCII
AR8ISO8859P6	ISO 8859-6 ラテン語 / アラビア語	SB、ASCII
EL8ISO8859P7	ISO 8859-7 ラテン語 / ギリシャ語	SB、ASCII
IW8ISO8859P8	ISO 8859-8 ラテン語 / ヘブライ語	SB、ASCII
NE8ISO8859P10	ISO 8859-10 北欧	SB、ASCII
WE8ISO8859P15	ISO 8859-15 西欧	SB、ASCII、EURO
LA8ISO6937	ISO 6937 8 ビット・テキスト通信用コード化キャラクタ・セット	SB、ASCII
IW7IS960	イスラエル標準 960 7 ビット・ラテン語 / ヘブライ語	SB
AR8ARABICMAC	Mac サーバー 8 ビット・ラテン語 / アラビア語	SB
EE8MACCE	Mac クライアント 8 ビット中欧	SB
EE8MACCROATIAN	Mac クライアント 8 ビット・クロアチア語	SB
WE8MACROMAN8	Mac クライアント 8 ビット拡張 Roman8 西欧	SB
EL8MACGREEK	Mac クライアント 8 ビット・ギリシャ語	SB
IS8MACICELANDIC	Mac クライアント 8 ビット・アイスランド語	SB
CL8MACCYRILLIC	Mac クライアント 8 ビット・ラテン語 / キリル文字	SB
AR8ARABICMACS	Mac サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII
EE8MACCES	Mac サーバー 8 ビット中欧	SB、ASCII

表 A-5 ヨーロッパ地域言語のキャラクタ・セット

名前	説明	コメント
EE8MACCROATIANS	Mac サーバー 8 ビット・クロアチア語	SB、ASCII
WE8MACROMAN8S	Mac サーバー 8 ビット拡張 Roman8 西欧	SB、ASCII
CL8MACCYRILLICS	Mac サーバー 8 ビット・ラテン語 / キリル文字	SB、ASCII
EL8MACGREEKS	Mac サーバー 8 ビット・ギリシャ語	SB、ASCII
IS8MACICELANDICS	Mac サーバー 8 ビット・アイスランド語	SB
BG8MSWIN	MS Windows 8 ビット・ブルガリア・キリル文字	SB、ASCII
LT8MSWIN921	MS Windows コード・ページ 921 8 ビット・リトアニア語	SB、ASCII
ET8MSWIN923	MS Windows コード・ページ 923 8 ビット・エストニア語	SB、ASCII
EE8MSWIN1250	MS Windows コード・ページ 1250 8 ビット東欧	SB、ASCII、EURO
CL8MSWIN1251	MS Windows コード・ページ 1251 8 ビット・ラテン語 / キリル文字	SB、ASCII、EURO
WE8MSWIN1252	MS Windows コード・ページ 1252 8 ビット西欧	SB、ASCII、EURO
EL8MSWIN1253	MS Windows コード・ページ 1253 8 ビット・ラテン語 / ギリシャ語	SB、ASCII、EURO
BLT8MSWIN1257	MS Windows コード・ページ 1257 8 ビット・バルト語	SB、ASCII、EURO
BLT8CP921	ラトビア標準 LVS8-92(1) Windows/Unix 8 ビット・バルト語	SB、ASCII
LV8PC8LR	ラトビア・バージョン IBM-PC コード・ページ 866 8 ビット・ラテン語 / キリル文字	SB、ASCII
WE8NCR4970	NCR 4970 8 ビット西欧	SB、ASCII
WE8NEXTSTEP	NeXTSTEP ポストスクリプト 8 ビット西欧	SB、ASCII
CL8KOI8R	RELCOM インターネット標準 8 ビット・ラテン語 / キリル文字	SB、ASCII
US8BS2000	Siemens 9750-62 EBCDIC 8 ビット米語	SB
DK8BS2000	Siemens 9750-62 EBCDIC 8 ビット・デンマーク語	SB
F8BS2000	Siemens 9750-62 EBCDIC 8 ビット・フランス語	SB
D8BS2000	Siemens 9750-62 EBCDIC 8 ビット・ドイツ語	SB
E8BS2000	Siemens 9750-62 EBCDIC 8 ビット・スペイン語	SB
S8BS2000	Siemens 9750-62 EBCDIC 8 ビット・スウェーデン語	SB
DK7SIEMENS9780X	Siemens 97801/97808 7 ビット・デンマーク語	SB

表 A-5 ヨーロッパ地域言語のキャラクタ・セット

名前	説明	コメント
F7SIEMENS9780X	Siemens 97801/97808 7 ビット・フランス語	SB
D7SIEMENS9780X	Siemens 97801/97808 7 ビット・ドイツ語	SB
I7SIEMENS9780X	Siemens 97801/97808 7 ビット・イタリア語	SB
N7SIEMENS9780X	Siemens 97801/97808 7 ビット・ノルウェー語	SB
E7SIEMENS9780X	Siemens 97801/97808 7 ビット・スペイン語	SB
S7SIEMENS9780X	Siemens 97801/97808 7 ビット・スウェーデン語	SB
WE8BS2000	Siemens EBCDIC.DF.04 8 ビット西欧	SB
CL8BS2000	Siemens EBCDIC.EHC.LC 8 ビット・キリル文字	SB
AL24UTFFSS	Unicode 1.1 UTF-8 ユニバーサル・キャラクタ・セット	MB、ASCII、EURO
UTF8	Unicode 2.0 UTF-8 ユニバーサル・キャラクタ・セット	MB、ASCII、EURO

中東地域言語のキャラクタ・セット

表 A-6 に、中東地域の言語をサポートしている Oracle キャラクタ・セットを示します。

表 A-6 中東地域のキャラクタ・セット

名前	説明	コメント
AR8APTEC715	APTEC 715 サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII
AR8ASMO708PLUS	ASMO 拡張 708 Plus8 ビット・ラテン語 / アラビア語	SB、ASCII
AR8ASMO8X	ASMO 拡張 708 8 ビット・ラテン語 / アラビア語	SB、ASCII
AR8ADOS710	アラビア語 MS-DOS 710 サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII
AR8ADOS720	アラビア語 MS-DOS 720 サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII
TR7DEC	DEC VT100 7 ビット・トルコ語	SB
TR8DEC	DEC 8 ビット・トルコ語	SB
WE8EBCDIC37C	EBCDIC コード・ページ 37 8 ビット Oracle/c	SB
IW8EBCDIC424	EBCDIC コード・ページ 424 8 ビット・ラテン語 / ヘブライ語	SB
WE8EBCDIC500C	EBCDIC コード・ページ 37 8 ビット Oracle/c	SB
IW8EBCDIC1086	EBCDIC コード・ページ 1086 8 ビット・ヘブライ語	SB

表 A-6 中東地域のキャラクタ・セット

名前	説明	コメント
AR8EBCDICX	EBCDIC XBASIC サーバー 8 ビット・ラテン語 / アラビア語	SB
TR8EBCDIC1026	EBCDIC コード・ページ 1026 8 ビット・トルコ語	SB
TR8PC857	IBM-PC コード・ページ 857 8 ビット・トルコ語	SB、ASCII
IW8PC1507	IBM-PC コード・ページ 1507/862 8 ビット・ラテン語 / ヘブライ語	SB、ASCII
AR8ISO8859P6	ISO 8859-6 ラテン語 / アラビア語	SB、ASCII
IW8ISO8859P8	ISO 8859-8 ラテン語 / ヘブライ語	SB、ASCII
WE8ISO8859P9	ISO 8859-9 西欧およびトルコ語	SB、ASCII
LA8ISO6937	ISO 6937 8 ビット・テキスト通信用コード化キャラクタ・セット	SB、ASCII
IW7IS960	イスラエル標準 960 7 ビット・ラテン語 / ヘブライ語	SB
IW8MACHEBREW	Mac クライアント 8 ビット・ヘブライ語	SB
AR8ARABICMAC	Mac クライアント 8 ビット・ラテン語 / アラビア語	SB
TR8MACTURKISH	Mac クライアント 8 ビット・トルコ語	SB
IW8MACHEBREWS	Mac サーバー 8 ビット・ヘブライ語	SB、ASCII
AR8ARABICMACS	Mac サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII
TR8MACTURKISHS	Mac サーバー 8 ビット・トルコ語	SB、ASCII
TR8MSWIN1254	MS Windows コード・ページ 1254 8 ビット・トルコ語	SB、ASCII、EURO
IW8MSWIN1255	MS Windows コード・ページ 1255 8 ビット・ラテン語 / ヘブライ語	SB、ASCII、EURO
AR8MSWIN1256	MS Windows コード・ページ 1256 8 ビット・ラテン語 / アラビア語	SB、ASCII、EURO
IN8ISCII	複数スクリプト・インド標準 8 ビット・ラテン / インド言語	SB
AR8MUSSAD768	Mussa'd Alarabi/2 768 サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII
AR8NAFITHA711	Nafitha 拡張 711 サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII
AR8NAFITHA721	Nafitha International 721 サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII
AR8SAKHR706	SAKHR 706 サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII
AR8SAKHR707	SAKHR 707 サーバー 8 ビット・ラテン語 / アラビア語	SB、ASCII

表 A-6 中東地域のキャラクタ・セット

名前	説明	コメント
WE8BS2000L5	Siemens EBCDIC.DF.04.L5 8 ビット西欧 / トルコ語	SB
AL24UTFFSS	Unicode 1.1 UTF-8 ユニバーサル・キャラクタ・セット	MB、ASCII、EURO
UTF8	Unicode 2.0 UTF-8 ユニバーサル・キャラクタ・セット	MB、ASCII、EURO

ユニバーサル・キャラクタ・セット

表 A-7 に、世界共通語をサポートしている Oracle キャラクタ・セットを示します。つまり、これらのキャラクタ・セットは、アジア、ヨーロッパおよび中東地域の言語に限らず、それらを含む世界中のすべての言語をサポートしようとするものです。

表 A-7 ユニバーサル・キャラクタ・セット

名前	説明	コメント
AL24UTFFSS	Unicode 1.1 UTF-8 ユニバーサル・キャラクタ・セット	MB、ASCII、EURO
UTF8	Unicode 2.0 UTF-8 ユニバーサル・キャラクタ・セット	MB、ASCII、EURO

注意 : Unicode 2.0 キャラクタ・セットは、Unicode 1.1 のかわりに使用されます。バージョン 1.1 とバージョン 2.0 の主な違いの 1 つは、11,172 個の韓国文字が再定義および追加されたことです。できる限り、最新バージョンの Unicode 規格を使用してください。Unicode 2.0 で現在サポートされている主な文字は、次のとおりです。

アラビア文字	グジャラート文字	ラテン文字
アルメニア文字	グルムキー文字	ラオ文字
ベンガル文字	漢字	マラヤーラム文字
ボポモフォ（Bopomofo）	ハングル文字	オリヤー文字
キリル文字	ヘブライ文字	タミル文字
デーヴァナーガリ文字	ひらがな	テルグ文字
グルジア文字	カンナダ文字	タイ文字
ギリシア文字	カタカナ	チベット文字

Unicode 規格の詳細は、http: / / www.unicode.org、または Unicode Consortium が定義している「Unicode Standard」を参照してください

言語の定義

言語の定義とは、特定の言語について言語上の事例を定義するものです。拡張言語定義は、言語について特別な言語上の事例を含みます。一般的に、拡張定義を使用して文字をソートすると、文字の ASCII 値でソートした場合と異なる結果が得られます。たとえば、*ch* と *ll* は、XSPANISH では 1 つの文字として扱われます。表 A-8 に、Oracle Server がサポートしている言語の定義を示します。

表 A-8 言語の定義

基本名	拡張名	特殊な事例
ARABIC	--	
ARABIC_MATCH	--	
ARABIC_ABJ_SORT	--	
ARABIC_ABJ_MATCH	--	
ASCII7	--	
BENGALI	--	
BULGARIAN	--	
CANADIAN FRENCH	--	
CATALAN	XCATALAN	æ、AE、ß
CROATIAN	XCROATIAN	D、L、N、d、l、n、ß
CZECH	XCZECH	ch、CH、Ch、ß
DANISH	XDANISH	A、ß、Å、å
DUTCH	XDUTCH	ij、IJ
EEC_EURO	--	
EEC_EUROPA3	--	
ESTONIAN	--	
FINNISH	--	
FRENCH	XFRENCH	
GERMAN	XGERMAN	ß
GERMAN_DIN	XGERMAN_DIN	ß、ä、ö、ü、Ä、Ö、Ü
GREEK	--	
HEBREW	--	

表 A-8 言語の定義

基本名	拡張名	特殊な事例
HUNGARIAN	XHUNGARIAN	cs、gy、ny、sz、ty、zs、ß、CS、Cs、GY、Gy、NY、Ny、SZ、Sz、TY、Ty、ZS、Zs
ICELANDIC	--	
INDONESIAN	--	
ITALIAN	--	
JAPANESE	--	
LATIN	--	
LATVIAN	--	
LITHUANIAN	--	
MALAY	--	
NORWEGIAN	--	
POLISH	--	
PUNCTUATION	XPUNCTUATION	
ROMANIAN	--	
RUSSIAN	--	
SLOVAK	XSLOVAK	dz、DZ、Dz、ß（*caron*）
SLOVENIAN	XSLOVENIAN	ß
SPANISH	XSPANISH	ch、ll、CH、Ch、LL、Ll
SWEDISH	--	
SWISS	XSWISS	ß
THAI_DICTIONARY	--	
THAI_TELEPHONE	--	
TURKISH	XTURKISH	æ、AE、ß
UKRAINIAN	--	
UNICODE_BINARY		
VIETNAMESE	--	
WEST_EUROPEAN	XWEST_EUROPEAN	ß

暦法

デフォルトでは、ほとんどの地域定義でグレゴリアン暦が使用されています。表 A-9 に、Oracle Server がサポートしている他の暦法を示します。

表 A-9 NLS がサポートしているカレンダ

名前	デフォルトの書式	デフォルトの書式で使用されるキャラクタ・セット
Japanese Imperial（日本の元号制）	EEYY"\307\257"MM"\267\356"DD"\306\374"	JA16EUC
ROC Official（台湾の公式の暦）	EEyy"\310\241"mm"\305\314"dd"\305\312"	ZHT32EUC
Thai Buddha（タイ仏教暦）	dd month EE yyyy	TH8TISASCII
Persian（ペルシャ暦）	DD Month YYYY	AR8ASMO8X
Arabic Hijrah（イスラム紀元）	DD Month YYYY	AR8ISO8859P6
English Hijrah（英語版イスラム紀元）	DD Month YYYY	AR8ISO8859P6

1998 年 3 月 20 日は、ROC Official（台湾の公式の暦）では次のように表示されます。

```
SQL> alter session set NLS_CALENDAR='ROC Official';

Session altered.

SQL> alter session set NLS_DATE_FORMAT =
  2  '"中華民國"YY"年"MM"月"DD"日"';

Session altered.

SQL> select sysdate from dual;

SYSDATE
--------------------
中華民國87年03月20日
```

1998 年 3 月 27 日は、Japanese Imperial（日本の元号制）では次のように表示されます。

```
SQL> alter session set NLS CALENDAR =
  2  'Japanese Imperial';

Session altered.

SQL> alter session set NLS DATE FORMAT=
  2  '"平成"YY"年"MM"月"DD"日"'

Session altered.

SQL> select sysdate from dual;

SYSDATE
----------------
平成10年03月27日
```

ユーロ記号をサポートしているキャラクタ・セット

表 A-10 に、ユーロ記号をサポートしているキャラクタ・セットを示します。

表 A-10 ユーロ記号を含むキャラクタ・セット

名前	説明	ユーロのコード値
D8EBCDIC1141	EBCDIC コード・ページ 1141 8 ビット・オーストリア・ドイツ語	0x9F
DK8EBCDIC1142	EBCDIC コード・ページ 1142 8 ビット・デンマーク語	0x5A
S8EBCDIC1142	EBCDIC コード・ページ 1143 8 ビット・スウェーデン語	0x5A
I8EBCDIC1144	EBCDIC コード・ページ 1144 8 ビット・イタリア語	0x9F
F8EBCDIC1147	EBCDIC コード・ページ 1147 8 ビット・フランス語	0x9F
WE8PC858	IBM-PC コード・ページ 858 8 ビット西欧	0xD5

表 A-10 ユーロ記号を含むキャラクタ・セット

名前	説明	ユーロのコード値
WE8ISO8859P15	ISO 8859-15 西欧	0xA4
EE8MSWIN1250	MS Windows コード・ページ 1250 8 ビット東欧	0x80
CL8MSWIN1251	MS Windows コード・ページ 1251 8 ビット・ラテン語 / キリル文字	0x88
WE8MSWIN1252	MS Windows コード・ページ 1252 8 ビット西欧	0x80
EL8MSWIN1253	MS Windows コード・ページ 1253 8 ビット・ラテン語 / ギリシャ語	0x80
TR8MSWIN1254	MS Windows コード・ページ 1254 8 ビット・トルコ語	0x80
BLT8MSWIN1257	MS Windows コード・ページ 1257 バルト語	0x80
VN8MSWIN1258	MS Windows コード・ページ 1253 8 ビット・ベトナム語	0xA0
TH8TISASCII	タイ工業規格 520-2533 - ASCII 8 ビット	0x80
AL24UTFFSS	Unicode 1.1 UTF-8 ユニバーサル・キャラクタ・セット	U+20AC
UTF8	Unicode 2.0 UTF-8 ユニバーサル・キャラクタ・セット	U+20AC

B

ロケール・データのカスタマイズ

NLS データ・オブジェクト・セットは、すべての Oracle 配布セットに含まれており、一部カスタマイズすることができます。

この付録では、次のトピックについて説明します。

カスタマイズ済みキャラクタ・セット

既存の Oracle キャラクタ・セットにユーザー定義文字を追加することによって、Oracle のキャラクタ・セット定義ファイルを拡張できます。

キャラクタ・セット情報とコード化方式はテキスト・ファイルに定義されています。これらのキャラクタ・セット定義のテキスト・ファイルにはキャラクタ・セットの説明があります。また、データベース管理者がキャラクタ・セットの変更または新規作成を容易に行えるように指定されています。すべての文字は Unicode 2.0 のコード・ポイントによって定義されています。つまり、それぞれの文字が Unicode 2.0 の文字コード値で定義されています。キャラクタ・セット間の変換は、中間フォームとして Unicode を使用して行われます。

キャラクタ・セット定義ファイルを作成したら、それを、実行時に動的にメモリーへロードされる、プラットフォーム固有のバイナリ・ファイルにコンパイルしなければなりません。この付録で説明されている NLS Data Installation Utility（lxinst）を使用することによって、キャラクタ・セット定義のテキスト・ファイルをバイナリ形式に変換してインストールし、それを NLS データ・オブジェクト・セットにマージできます。

この手順では、次の動作が必ずしも行われるわけではありません。

- ユーザー定義文字の入力

  ユーザー定義文字の入力は、入力メソッドまたは仮想キーボードを介して行われ、システムによって管理される必要があります。

- ユーザー定義文字の表示

  ユーザー定義文字の表示は、システムまたはアプリケーション（あるいはその両方）によって管理される必要があります。表示する場合は、新しいフォントの指定が必要な場合があります。多くのベンダーがフォント・エディタをサポートしています。フォントを新規作成したら、ご使用のシステムにインストールし、アプリケーション・プログラムからアクセスできるようにする必要があります。

- ユーザー定義文字のソート

  ユーザー定義文字のソートは、サポートされていません。正確には、カスタマイズしたキャラクタ・セットのソートはいずれも現在サポートされていません。ただし、バイナリまたは言語ソートは選択できます。言語ソートの場合、事前定義済みの Oracle 言語ソートに限り使用できます。

キャラクタ・セット定義ファイル

キャラクタ・セット情報およびコード化方式はテキスト・ファイル（接尾辞は ".nlt"）に定義されています。キャラクタ・セット定義のテキスト・ファイル（*.nlt ファイル）にはキャラクタ・セットの記述があります。また、データベース管理者がキャラクタ・セットの変更または新規作成を容易に行えるようにわかりやすい形式で指定されています。すべての文字は Unicode 2.0 のコード・ポイントによって定義されています。つまり、それぞれの文字が Unicode 2.0 の文字コード値で定義されています。

キャラクタ・セット間の変換は、中間フォームとして Unicode を使用して行われます。次に、カスタマイズ済みキャラクタ・セット定義ファイルの形式テンプレートの例を示します。

カスタマイズ済みキャラクタ・セット定義ファイルの形式テンプレート

```
# The following is a template of a customized character set definition file.
# You may use this template to create a user-defined character set or copy
# and modify an existing one. The convention used for naming character
# set definition (.nlt) files is in the format: lx2dddd.nlt, where
#                    dddd = 4 digit character set ID in hex
# All letters in the definition file are case-insensitive.

# Version number: specify the current loadable data version.
VERSION = <x.x.x.x.x>

# The following is the body of the definition file
DEFINE character_set
```

```
# Oracle supports a feature called 'base_char_set'. It allows you
# to extend an existing character set based on an existing Oracle supported
# standard character set. Generally, you may only need to edit the
# following fields:

# Name and ID of the character set are required for any character sets.

# Character set name must be specified in a double quoted string.
# Rules for choosing a character set name:
#          - Cannot use a character set name that is already in use. (Each
#            character set must be assigned a unique character set name).
#          - Must consist of single-byte ASCII or EBCDIC characters only
#            (single-byte compiler character set).
#          - Cannot contain multibyte characters.
#          - Maximum length of 30 characters.
#          - Must start with an alphabetic character.
#          - Composed of alphanumeric characters only (e.g. no periods,
#            dashes, underscore characters allowed)
#          - The name is case-insensitive.
# To register a unique character set name, send mail to
# nlsreg@us.oracle.com.
   name = <text_string>

# Character set ID is specified as an integer value.
# Rules for choosing a character set ID:
#         - Cannot use a character set ID that is already in use. (Each
#           character set must be assigned a unique character set ID.)
#         - Must be in the decimal range of 10000-20000
#         - Character set IDs must be registered with Oracle to receive a
#           uniquely assigned character set ID number.
# To register a unique character set ID, send mail to nlsreg@us.oracle.com.
   id   = <integer>

# The "base_char_set" feature allows users to define the base character set in   # a
new character set definition file.
# The new character set will inherit all definitions from the base
# character set, therefore, the user only needs to add the customized data
# into the new character set definition file.

# The syntax of the base character set is:
#       base_char_set = <id> | <name>

#        - <id> or <name> should be a valid Oracle NLS character set id or name.
# Example is: base_char_set = "JA16EUC" or base_char_set = 830
base_char_set = <id> | <name>

# If you use base_char_set feature, remember you need to copy your base
```

```
# character set definition file (text or binary format) from $ORA_NLS33
# into the working directory specified by $ORANLS so that the new character
# set can inherit the definition from the base character set.
# Example:
#  %cp $ORA_NLS33/lx2033e.nlt $ORANLS
# or
#  %cp $ORA_NLS33/lx*33e.nlb $ORANLS


# Character data is defined as a list of  <char_value>:<unicode_value>
# pairs. <char_value> is a hex number specifying the complete character
# value in this character set (e.g. 0xa1b1), while <unicode_value> is a
# 16-bit hex number specifying its corresponding Unicode 2.0 character
# value.
# Alternatively, a range of characters can be specified with a corresponding
# range of Unicode values. Each successive character in the
# <start_char>-<end_char> range will be assigned to each successive
# character in the <start_unicode>-<end_unicode> range. There must be
# an equal number of characters in each range.
# User-defined characters must be assigned to characters in Unicode's
# private use area, and in particular the range 0xe000 to 0xf4ff. The
# remaining 1024 characters in the private use area are reserved for Oracle
# private use.
# If you already defined "base_char_set", you only need to add the
# customized character set mappings.
   character_data = {
<char_value>:<unicode_value>,
<start_char>-<end_char>:<start_unicode>-<end_unicode>,
...
   }

# A character classification list is used to specify the type of characters.
# Valid values:
# UPPER LOWER DIGIT SPACE PUNCTUATION CONTROL
#            HEX_DIGIT LETTER PRINTABLE
# You only need to add customized characters' classification if you defined
# base_char_set.
classification = {
<char_value> = { UPPER, LOWER, DIGIT,
       SPACE, PUNCTUATION, CONTROL,
       HEX_DIGIT, LETTER, PRINTABLE },
...
   }

# Lower-to-Upper case character relationships are defined as pairs, where
# the first specifies the value of a character in this character set and the
# second specifies its uppercase value in this character set.  You may add
```

```
# the customized case mapping only if needed.
   uppercase = {
<char_value>:<upper_char_value>,
<start_char>-<end_char>:<start_upper>-<end_upper>,
...
   }

# Upper-to-Lower case character relationships are defined as pairs, where
# the first specifies the value of a character in this character set and the
# second specifies its lowercase value in this character set. You may add
# the customized case mapping only if needed.
   lowercase = {
<char_value>:<lower_char_value>,
<start_char>-<end_char>:<start_lower>-<end_lower>,
...
   }

# There are a lot of other fields in an Oracle character set definition file.
# Presumably, you will only need the above fields, at most.

ENDDEFINE character_set
```

キャラクタ・セットのカスタマイズの例

この項では、キャラクタ・セットを新規作成する場合に必要な手順を例をあげて説明します。この例では、Oracle の JA16EUC キャラクタ・セットに基づいたキャラクタ・セットを新規作成し、これにいくつかのユーザー定義文字を追加します。

手順 1. 新規キャラクタ・セットの名前および ID を登録する

キャラクタ・セットの名前および ID が一意になるように、キャラクタ・セット名を Oracle に登録し、一意に割り当てられたキャラクタ・セット ID を受け取ります。キャラクタ・セット名と ID の登録要求は、次のアドレスに送信してください。

```
nlsreg@us.oracle.com
```

注意 : キャラクタ・セット名と ID が一意でない場合、キャラクタ・セット間の非互換またはデータ損失の恐れが生じます。

キャラクタ・セット名には、次の制限事項があります。

- すでに使用中のキャラクタ・セット名は指定できません。（キャラクタ・セットごとに、一意のキャラクタ・セット名を割り当てなければなりません）
- シングルバイトの ASCII 文字または EBCDIC 文字（シングルバイト・コンパイラ・キャラクタ・セット）のみで構成する必要があります。
- 最大長は 30 文字です。
- 先頭はアルファベット文字である必要があります。
- 英数字のみで構成されていなければなりません。（ピリオド、ダッシュ、アンダースコア文字などは使用できません）
- 大文字と小文字は区別されません。

キャラクタ・セット ID を選択する場合の規則は、次のとおりです。

- すでに使用中のキャラクタ・セット ID は指定できません。（キャラクタ・セットごとに、一意のキャラクタ・セット ID を割り当てなければなりません）
- 10 進数で 10000 〜 20000 の範囲（16 進数では 0x2710 〜 0x3a98 の範囲）でなければなりません。

既存の Oracle のキャラクタ・セットから導出されたキャラクタ・セットの場合、次のキャラクタ・セット命名規則を使用することをお薦めします。

```
<Oracle_character_set_name><organization_name>EXT<version>
```

次に例を示します。

Sun Microsystems という会社が JA16EUC キャラクタ・セットにユーザー定義文字を追加した場合、適切なキャラクタ・セット名は次のようになります。

```
JA16EUCSUNWEXT1
```

この場合、次のような意味になります。

JA16EUC	Oracle が定義したキャラクタ・セット名です。
SUNW	会社名（Sun Microsystems の株式取引上の略称）を表します。
EXT	JA16EUC キャラクタ・セットに対する拡張要素であることを示します。
1	バージョンを示します。

この例と以降のすべての手順では、キャラクタ・セット ID を 10000（16 進数値で 0x2710）とします。

手順 2. NLS テキスト・ブート・ファイルを作成する

NLS バイナリ・ブート・ファイルには、データベースにロードされる NLS データ・オブジェクトが示されています。したがって、キャラクタ・セットが新しく作成されるたびにバイナリ・ブート・ファイルを更新する必要があります。バイナリ・ブート・ファイルを更新するには、まずテキスト・ブート・ファイル lx0boot.nlt に設定する新規キャラクタ・セット用のエントリを作成する必要があります。

NLS ブート・ファイル形式

```
# The following is a template for an Oracle NLS boot file.

# Version number specifies the current loadable data version.
VERSION=<x.x.x.x.x>

# List the character set names and IDs that will be merged into the existing
# system boot file using the $ORACLE_HOME/bin/lxinst utility.
#
CHARACTER_SET
<name> <id>
<name> <id>
...
```

次に例を示します。

テキスト・ブート・ファイル（lx0boot.nlt）を作業ディレクトリに作成します。

```
% vi /tmp/lx0boot.nlt
```

JA16EUCSUNWEXT1 を追加するには、次のように設定します。

```
VERSION=2.1.0.0.0

CHARACTER_SET
"JA16EUCSUNWEXT1"  10000
```

バージョン・ナンバーは、Oracle リリースに基づいています。最新のバージョン・ナンバーについては、既存の lx2*.nlt ファイルに記述されているバージョン・ナンバーを参照してください。

複数のユーザー定義キャラクタ・セットを 1 つの lx0boot.nlt ファイルに記述できます。たとえば、次のように指定します。

```
VERSION=2.1.0.0.0

CHARACTER_SET
"JA16EUCSUNWEXT1"  10000
```

```
"ZH16EUCSUNWEXT1"  10001
```

手順 3. キャラクタ・セット定義ファイル（lx2*dddd*.nlt）を作成する
キャラクタ・セット定義ファイル（.nlt）の命名規則には、書式 lx2*dddd*.nlt を使用します。この場合、*dddd* は、16 進数の 4 桁のキャラクタ・セット ID です。

キャラクタ・セット定義ファイルを編集する際には、次の点に注意してください。

- 既存の Oracle キャラクタ・セットに対してのみ拡張（文字を追加）できます。
- 既存の文字を再マップしてはいけません。
- すべての文字のマッピングは一意でなければなりません。
- 1 文字を複数の文字コードにマッピングすることはできません。
- 複数の文字コードを 1 つの文字にマッピングすることはできません。
- 新規文字は、Unicode のプライベートな使用範囲（e000 ～ f4ff）にマップする必要があります。（実際の Unicode 2.0 のプライベート使用範囲は e000 ～ f8ff ですが、f500 ～ f8ff は Oracle 固有のプライベート使用のために予約済みです。）
- キャラクタ・セット定義ファイルでは、1 行の最大文字数は 80 文字です。

「BASE_CHAR_SET」という機能があります。これによって、カスタマイズ済みキャラクタ・セットが簡単にサポートできます。既存の Oracle キャラクタ・セットを拡張している場合は「BASE_CHAR_SET」機能を使用できます。この機能によって、新しいキャラクタ・セットは基本キャラクタ・セットからすべての定義を継承するため、ユーザーは独自のカスタマイズ済みキャラクタ・セットのデータを追加するだけでよくなります。

次に例を示します。

JA16EUC キャラクタ・セットを拡張し、これに新規のカスタマイズ済みキャラクタ・セット・データを追加したと仮定します。

手順 1 で指定したキャラクタ・セット ID 10000 に基づいて、新規のキャラクタ・セット定義ファイルに、lx2*2710*.nlt（キャラクタ・セット ID の 16 進数値 0x2710 に基づいて）と名前を付けます。

この例では、作業ディレクトリとして **/tmp** を使用します。エディタを使用して、新規キャラクタ・セット定義ファイルを編集します。

```
% vi /tmp/lx22710.nlt
VERSION = 2.1.0.0.0

DEFINE character_set
  name = "JA16EUCSUNWEXT1"
  id = 10000
  base_char_set = 830
  character_data = {
     0x9a41 : 0xe001,
     0x9a42 : 0xe002,
```

```
  }
  classification = {
     0x9a41 = { LETTER, LOWER },
     0x9a42 = { LETTER, UPPER },
  }
  uppercase = {
     0x9a41 : 0x9a42,
  }
  lowercase = {
     0x9a42 : 0x9a41,
  }
ENDDEFINE character_set
```

キャラクタ・セット定義ファイルの形式については、B-2 ページ「カスタマイズ済みキャラクタ・セット定義ファイルの形式テンプレート」を参照してください。少なくとも、キャラクタ・セット名、キャラクタ・セット ID および基本キャラクタ・セットを設定し、カスタマイズ済みキャラクタ・セット・データと分類フィールドを追加する必要があります。

手順 4. NLS バイナリ・ブート・ファイルをバックアップする

.nlb ファイルを生成しインストールする前に、NLS インストレーション・ブート・ファイル（lx0boot.nlb）および NLS システム・ブート・ファイル（lx1boot.nlb）を ORA_NLS33 ディレクトリにバックアップしておくことをお薦めします。

```
% cd $ORA_NLS33
% cp lx0boot.nlb lx0boot.nlb.orig
% cp lx1boot.nlb lx1boot.nlb.orig
```

手順 5. .nlb ファイルを生成しインストールする

この時点で、新規の .nlb ファイルを生成しインストールする準備が整います。.nlb ファイルはプラットフォームに依存するので、必ずプラットフォームごとにファイルを生成しなおしてください。また、サーバーとクライアントの両方にこれらのファイルをインストールする必要があります。

lxinst ユーティリティを使用して、バイナリ・キャラクタ定義ファイル（lx2dddd.nlb）を作成し、NLS ブート・ファイル（lx*boot.nlb）を更新します。

次に例を示します。

lxinst ユーティリティでは、既存のシステム・ブート・ファイルを利用します。したがって、既存のバイナリ・システム・ブート・ファイルを SYSDIR で指定したディレクトリにコピーします。この例では、SYSDIR に作業ディレクトリ（/tmp）が指定されています。

```
% cp lx1boot.nlb /tmp
```

手順 2 と 3 で作成した新規のキャラクタ・セット定義ファイル（lx22710.nlt）、および新規キャラクタ・セットのエントリを含むテキスト・ブート・ファイル（lx0boot.nlt）は、ORANLS で指定したディレクトリに存在する必要があります。この例では、ORANLS には /tmp が指定されています。また、JA16EUC（ID 830、16 進数値で 033e）は「BASE_CHAR_SET」として定義されているので、基本定義ファイルのテキスト形式（lx2033e.nlt）またはバイナリ形式（lx*033e.nlb）もディレクトリ ORANLS 内に存在する必要があります。そうすれば、新規キャラクタ・セットはすべての定義をそのファイルから継承できます。

```
% cp lx2033e.nlt /tmp
```

または

```
% cp lx*033e.nlb /tmp
```

lxinst ユーティリティを使用して、ORANLS で指定したディレクトリにバイナリのキャラクタ・セット定義ファイル（lx22710.nlb）を生成し、DESTDIR で指定したディレクトリに更新済みのバイナリ・ブート・ファイル（lx1boot.nlb）を生成します。この例では、ORANLS、SYSDIR および DESTDIR のすべてに /tmp が定義されています。

```
% $ORACLE_HOME/bin/lxinst oranls=/tmp sysdir=/tmp destdir=/tmp
```

次に、新しく生成したバイナリ・ブート・ファイル（lx1boot.nlb）を ORA_NLS33 ディレクトリにインストールします。

```
% cp /tmp/lx1boot.nlb $ORA_NLS33/lx1boot.nlb
```

最後に、新規のキャラクタ・セット定義ファイル lx2*.nlb を ORA_NLS33 ディレクトリにインストールします。lx5*.nlb または lx6*.nlb、あるいはその両方が存在する場合は、これらもインストールします。

```
% cp /tmp/lx22710.nlb $ORA_NLS33
% cp /tmp/lx52710.nlb $ORA_NLS33
% cp /tmp/lx62710.nlb $ORA_NLS33
```

手順 6. プラットフォームごとに作業を繰り返す
.nlb ファイルはプラットフォーム固有のバイナリ・ファイルであるため、手順 5 の作業をハードウェアのプラットフォームごとに繰り返す必要があります。また、新しいキャラクタ・セットの認識が必要なすべてのシステムに対しても、手順 5 の作業を繰り返し行ってください。したがって、新しい .nlb ファイルをサーバー・マシンとクライアント・マシンの両方でコンパイルしインストールしてください。

手順 7. 新規キャラクタ・セットを使用してデータベースを作成する
.nlb ファイルをインストールした後、NLS データのロードを初期化するためにデータベース・サーバーをいったんシャットダウンしてから再起動する必要があります。

データベース・サーバーのバックアップをとった後、新たに作成したキャラクタ・セットを使用して新規データベースを作成してください。

クライアント側で新規キャラクタ・セットを使用するには、単にクライアント（Enterprise Manager、SQL*Plus など）を終了し、.nlb ファイルのインストール後に再起動するだけです。

カスタマイズ済みカレンダ

概要

グレゴリオ暦の他に多数のカレンダがサポートされています。これらのカレンダはすべて、NLS に直接リンクしたデータで定義されていますが、一部のカレンダについては、将来、元号（日本の元号制の場合）または偏差日（太陰暦の場合）の追加が必要になるものもあります。新規リリースを待たずにこれらを追加するために、追加する年号または偏差日を外部ファイルに定義できます。外部ファイルは、カレンダ機能を実行するときに自動的にロードされます。

カレンダのデータは、まずテキスト形式の定義ファイルに定義します。このファイルは使用する前に、バイナリ形式に変換する必要があります。この変換を行うには、この項で説明する Calendar Utility を使用します。

NLS Calendar Utility

構文

Calendar Utility は、コマンド行から次のように直接起動します。

```
LXEGEN
```

パラメータはありません。

使用方法

Calendar Utility は、入力としてテキスト形式の定義ファイルを使用します。ファイルの名前と位置は、プラットフォームに依存する値でハードコードされています。UNIX プラットフォームでは、ファイルの名前は lxecal.nlb で、その位置は **$ORACLE_HOME/ocommon/nls** です。カレンダ定義ファイルのサンプルは、配布製品に含まれています。

注意 : ファイルの場所は、プラットフォームによって異なります。ご使用のシステムでのファイルの場所については、プラットフォーム固有の Oracle ドキュメントを参照してください。

lxegen 実行可能ファイルは、適切な形式のカレンダ・データを含むバイナリ・ファイルを出力として生成します。出力ファイル名もまた、プラットフォームに依存する値でハードコードされています。たとえば、UNIX の場合、イスラム紀元暦の偏差日数を定義するのであれば、ファイル名は lxecal.nlb となります。出力ファイルは、テキスト形式ファイルと同じディレクトリに生成され、既存のファイルは上書きされます。

バイナリ・ファイルが生成されると、そのファイルはシステムの初期化中に自動的にロードされます。このときファイルはハードコードされた名前と位置で検索されるので、このファイルを移動したり改名したりしないでください。

ユーティリティ

Oracle Server では、NLS データのメンテナンス用に次の 3 つのユーティリティが用意されています。

NLS Data Installation Utility（lxinst）	バイナリ形式データ・オブジェクトを、そのテキスト形式バージョンから生成します。NLS データ更新を受信する、または独自のデータ・オブジェクトを作成する場合に使用します。
NLS Calendar Utility（lxegen）	適切な書式を使用してカレンダ・データに対するバイナリ・ファイルを生成します。
NLS Configuration Utility（lxbcnf）	ユーザー・ブート・ファイルを作成し、編集します。

NLS Data Installation Utility

概要

Oracle の配布セットには、NLS データ・オブジェクトのデフォルトのセットが含まれています。一部の NLS データ・オブジェクトはカスタマイズできます。たとえば、Oracle8*i* では、Oracle のキャラクタ・セット定義ファイルを拡張してユーザー定義文字を追加できます。このような NLS 定義ファイルは、バイナリ形式に変換してから既存の NLS オブジェクト・セットにマージする必要があります。この作業を行うには、ここで説明する NLS Data Installation Utility を使用します。

バイナリ・オブジェクト・ファイルの他に、ブート・ファイルも NLS Data Installation Utility によって生成されます。ブート・ファイルは、ロードする必要があるすべての NLS オブジェクトを識別してその位置を調べるために、それぞれのモジュールで使用されます。

ブート・ファイルの配布およびユーザーの構成を容易にするために、次の 3 種類のブート・ファイルが定義されています。

インストレーション・ブート・ファイル	配布セットの一部として含まれているブート・ファイル。
システム・ブート・ファイル	NLS Data Installation Utility が生成するブート・ファイル。NLS オブジェクトをロードします。システム・ブート・ファイルがすでにインストールされている場合、その内容は、オブジェクトの生成中に新しいシステム・ブート・ファイルの内容とマージされます。
ユーザー・ブート・ファイル	システム・ブート・ファイル情報のサブセットを含むブート・ファイル。このファイルの生成方法の詳細は、「NLS Configuration Utility」を参照してください。

構文

NLS Data Installation Utility は、次の構文を使用してコマンド行から起動します。

```
LXINST [ORANLS=pathname] [SYSDIR=pathname] [DESTDIR=pathname] [HELP=[yes | no]]
[WARNING=[0 | 1 | 2 | 3]]
```

ここでは、次のような意味になります。

ORANLS=pathname	テキスト形式のブート・ファイルとオブジェクト・ファイルを検索する場所、およびバイナリ形式の新しいブート・ファイルとオブジェクト・ファイルを格納する場所を指定します。指定しなかった場合、NLS Installation Utility は環境変数 ORA_NLS33（または、ご使用のオペレーティング・システムでの同等の変数）の値を使用します。両方が指定されている場合は、環境変数よりもコマンド行で指定したパラメータが優先されます。どちらも指定されていない場合は、NLS Installation Utility はエラーを出力して終了します。
SYSDIR=pathname	既存のシステム・ブート・ファイルを検索する場所を指定します。指定しなかった場合、NLS Installation Utility は初期化ファイル・パラメータ ORANLS に指定されたディレクトリを使用します。既存のシステム・ブート・ファイルが存在しない場合、または NLS Installation Utility によってファイルが検出されなかった場合、新しいファイルが生成され、適切なディレクトリにコピーされます。
DESTDIR=pathname	新しい（マージされた）システム・ブート・ファイルを格納する場所を指定します。指定しなかった場合、NLS Installation Utility は初期化ファイル・パラメータ ORANLS に指定されたディレクトリを使用します。このディレクトリに存在するシステム・ブート・ファイルはすべて上書きされるので、最初にファイルをバックアップしておいてください。
HELP=[yes | no]	「yes」を指定した場合、NLS Installation Utility の構文を説明するヘルプ・メッセージが表示されます。

[WARNING= [0 | 1 | 2 | 3]]	「0」を指定すると、警告メッセージは表示されません。「1」を指定すると、レベル 1 のメッセージがすべて表示されます。「2」を指定すると、レベル 2 およびレベル 1 のメッセージがすべて表示されます。「3」を指定すると、レベル 3、レベル 2 およびレベル 1 のメッセージがすべて表示されます。

戻り値

lxinst を実行すると、次の戻り値が返ります。

0	バイナリのブート・ファイルとオブジェクト・ファイルの生成、およびインストレーション・ブート・ファイルとシステム・ブート・ファイルのマージが正常に終了しました。
1	インストレーションが失敗しました。NLS Installation Utility は、問題を示すエラー・メッセージを出力して終了します。

使用方法

lxinst を使用してカスタマイズしたキャラクタ・セットをインストールするには、次の作業を行います。

- 新規データ・オブジェクトへの参照を含むテキスト形式のブート・ファイル（lx0boot.nlt）を作成する。
  - データ・オブジェクトは、ブート・ファイル内で参照されている場合のみ生成できる。
  - キャラクタ・セットのオブジェクト型のみ生成できる。
- テキスト形式のデータ・オブジェクト・ファイルを新規作成する。命名規則の詳細は、B-15 ページの「データ・オブジェクト・ファイル名」を参照してください。

  注意 : 配布セットには、キャラクタ・セット定義のデモ・ファイルが含まれています。このファイルは、参照用またはテンプレートとして使用できます。UNIX ベースのシステムでは、このファイルは **$ORACLE_HOME/demo/*.nlt** にあります。
- 前述の説明に従って（適切なパラメータを使用して）lxinst を起動し、バイナリ・データ・オブジェクト・ファイルを新規作成する。これらのファイルは、ORANLS で指定したディレクトリに生成されます。
  - lxinst によって、インストレーション・ブート・ファイルおよびシステム・ブート・ファイルの両方を新規作成することもできる。以前に NLS インストレーションを実行していて、既存の情報をシステム・ブート・ファイル内の新しい情報とマージする場合は、既存のシステム・ブート・ファイルを SYSDIR で指定したディレクトリにコピーします。マージされた情報を含む新しいシステム・ブート・ファイルが、DESTDIR で指定したディレクトリに生成されます。

注意 : 上書きしたくない既存のファイルはすべて、通常どおりバックアップしておいてください。

オブジェクト型

現在、カスタマイズできるオブジェクト型は、キャラクタ・セットのみです。

オブジェクト ID

NLS データ・オブジェクトは、オブジェクト ID の数値によって一意に識別されます。ID にゼロまたは負の値を使用することはできません。

通常、オブジェクト ID を 10000 ～ 20000 の範囲内で指定する限り、新しいオブジェクトを定義できます。

> **警告 :** キャラクタ・セットを新規作成する場合は、nlsreg@us.oracle.com 宛てに電子メールを送信してオラクル社に登録する必要があります。これによって、ご使用のキャラクタ・セットの名前および ID は確実に一意になります。

オブジェクト名

オブジェクト名としては、次の文字セットのみが使用できます。

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_- and <space>
```

オブジェクト名の先頭はアルファベット文字でなければなりません。言語、地域およびキャラクタ・セットの名前には、アンダースコア文字を使用できませんが、言語定義名には使用できます。オブジェクト名では、大文字と小文字は区別されません。オブジェクト名の最大サイズは 30 バイトです（NULL 終了文字を除く）。

データ・オブジェクト・ファイル名

オブジェクト・ファイル名はシステムに依存せず、一般的なブート・ファイルのエントリ情報から作成されます。

```
lxtdddd
```

ここでは、次のような意味になります。

t	1 桁のオブジェクト型（16 進）
dddd	4 桁のオブジェクト ID（16 進）

インストレーション・ブート・ファイル名は *lx*0BOOT、システム・ブート・ファイル名は *lx*1BOOT、ユーザー・ブート・ファイル名は *lx*2BOOT になります。ファイル拡張子は、テキスト形式ファイルの場合は .nlt、バイナリ・ファイルの場合は .nlb です。

次に例を示します。

lx22711.nlt	テキスト形式のキャラクタ・セット定義で、ID は 10001。
lx0boot.nlt	テキスト形式のインストレーション・ブート・ファイル。
lx1boot.nlb	バイナリのシステム・ブート・ファイル。
lx22711.nlb	バイナリのキャラクタ・セット定義で、ID は 10001。

NLS Configuration Utility

概要

インストレーション時に、使用可能な NLS オブジェクトはすべてシステム・ブート・ファイル内に格納され、参照されます。このファイルを使用して、使用可能な NLS データをロードします。

NLS Configuration Utility を使用すると、必要な NLS オブジェクトのみをロードするように、ご使用のブート・ファイルを構成することができます。これを行うには、ユーザー・ブート・ファイルを作成して、システム・ブート・ファイルのサブセットを入れます。カーネルによるデータのロードは、このユーザー・ブート・ファイルの内容に従って実行されます。

NLS Configuration Utility でユーザー・ブート・ファイルを構成する方法は 2 通りあります。1 つは、インストール済みのシステム・ブート・ファイルから NLS オブジェクトを選択して、新しいユーザー・ブート・ファイルに入れる方法です。もう 1 つは、既存のユーザー・ブート・ファイルからエントリを読み込み、不要なエントリがあればそれを削除して、残りのエントリを新しいユーザー・ブート・ファイルに保存する方法です。既存のブート・ファイルは RDBMS またはその他の Oracle Tool が使用している可能性があるため、既存のファイルを直接編集することはできません（つまり、ブート・ファイルのエントリが、既存のブート・ファイルに保存されることはありませ）。

また、NLS Configuration Utility を使用して、既存のユーザー・ブート・ファイルの整合性を調べることもできます。既存の NLS オブジェクトが何度も変更され、新しいシステム・ブート・ファイルのインストレーションによってユーザー・ブート・ファイルの内容が古くなってしまう場合があります。このような場合に、整合性チェックを行います。比較機能を使用することによって、ファイルの内容が古くなった場合には通知されるので、新しいユーザー・ブート・ファイルを作成することができます。

構文

NLS Configuration Utility は、次の構文を使用してコマンド行から起動します。

```
LXBCNF [ORANLS=pathname] [userbootdir=pathname] [DESTDIR=pathname]
[HELP=[yes | no]]
```

ここでは、次のような意味になります。

ORANLS=pathname	テキスト形式のブート・ファイルとオブジェクト・ファイルを検索する場所、およびバイナリ形式の新しいブート・ファイルとオブジェクト・ファイルを格納する場所を指定します。指定しなかった場合、NLS Configuration Utility は環境変数 ORA_NLS（または、ご使用のオペレーティング・システムでの同等の変数）の値を使用します。両方が指定されている場合は、環境変数よりもコマンド行で指定したパラメータが優先されます。どちらも指定されていない場合は、NLS Configuration Utility はエラーを出力して終了します。
SYSDIR=pathname	既存のシステム・ブート・ファイルを検索する場所を指定します。指定しなかった場合、NLS Configuration Utility は初期化ファイル・パラメータ ORANLS に指定されたディレクトリを使用します。既存のシステム・ブート・ファイルが存在しない場合、または NLS Configuration Utility によってファイルが検出されなかった場合、ファイルが新規作成されて適切なディレクトリにコピーまたは移動されます。
DESTDIR=pathname	新しい（マージされた）システム・ブート・ファイルを格納する場所を指定します。指定しなかった場合、NLS Configuration Utility は初期化ファイル・パラメータ ORANLS に指定されたディレクトリを使用します。このディレクトリに存在するシステム・ブート・ファイルはすべて上書きされるので、最初にファイルをバックアップしておいてください。
HELP=[yes | no]	「yes」を指定した場合、NLS Configuration Utility の構文を説明するヘルプ・メッセージが表示されます。

メニュー

NLS Configuration Utility を実行すると、次のトップレベル・メニューが表示されます。

- 「ファイル」メニュー
- 「編集」メニュー
- 「アクション」メニュー
- 「ウィンドウ」メニュー
- 「ヘルプ」メニュー

「ファイル」メニュー

「ファイル」メニューには、ファイル操作に関する選択項目があります。メニュー項目は、次のとおりです。

表 B-1 「ファイル」メニュー・オプション

メニュー項目	オプション	説明
システム・ブート・ファイル	開く	現行のシステム・ブート・ファイルをオープンします。システム・ブート・ファイルが正常に読み込まれると、メニュー項目「オープン」はすぐに淡色表示になります。また、システム・ブート・ファイルがオープンするまでは、他の機能を実行することはできません。
ユーザー・ブート・ファイル	新規	新しいユーザー・ブート・ファイルをオープンします。
	読込み	既存のユーザー・ブート・ファイルの内容を読み込みます。
	保存	新しいユーザー・ブート・ファイルに対する変更を保存します。
	回復	現在オープンしているユーザー・ブート・ファイルに対して、前回の「保存」以降に行われた変更を取り消します。
プリンタ選択		今回のリリースでは、インプリメントされていません。
ページ設定		今回のリリースでは、インプリメントされていません。
印刷		今回のリリースでは、インプリメントされていません。
終了		ファイルを終了します。

注意 : システム・ブート・ファイルがオープンされ読み込まれない限り、これらのメニュー項目はすべて淡色表示されたままです。つまり、使用可能なシステム・ブート・ファイル情報がないと、ユーザー・ブート・ファイルを作成することはできません。

新しいユーザー・ブート・ファイルを作成するために「新規」を選択すると、引き続き NLS オブジェクト（デフォルト）がただちに新しいファイル内に作成されます。

既存のユーザー・ブート・ファイル内容の「読込み」を選択した場合、読み込まれたエントリがシステム・ブート・ファイルのエントリと照合されます。ここでシステム・ブート・ファイルに存在しないエントリが検出された場合は、警告が表示され、そのエントリは組み込まれません。

「編集」メニュー

「編集」メニューには、NLS Configuration Utility のダイアログまたはウィンドウで入力した情報を編集するための選択項目があります。

「アクション」メニュー

「アクション」メニューには、ユーザー・ブート・ファイルの操作を実行するための選択項目があります。このメニューは、文字モードの NLS Configuration Utility のみで使用可能です。

項目コピー	システム・ブート・ファイルから選択した項目をユーザー・ブート・ファイルにコピーします。
項目削除	ユーザー・ブート・ファイルから選択した項目を削除します。

「ウィンドウ」メニュー

「ウィンドウ」メニューを使用すると、特定のウィンドウをアクティブ化したり、すでにオープンされているウィンドウにフォーカスを設定したりできます。新しいウィンドウがオープンされるたびに、そのウィンドウの名前が「ウィンドウ」メニューに自動的に追加されます。

NLS デフォルト	今回のリリースでは、インプリメントされていません。

「ヘルプ」メニュー

このメニューを使用すると、NLS Configuration Utility に関するさまざまなレベルのヘルプ情報を検索できます。

バージョン情報	NLS Configuration Utility のバージョン情報を表示します。
ヘルプ・システム	今回のリリースでは、インプリメントされていません。

C

廃止されたロケール・データ

Oracle ではたくさんのキャラクタ・セットが何度も改名されてきました。この付録では、次のトピックについて説明します。

- 廃止された NLS データ

廃止された NLS データ

Oracle Server リリース 7.2 よりも前のリリースでは、キャラクタ・セットが改名された場合、通常その変更後のリリースでは、旧名も新規名と一緒にサポートされていました。リリース 7.2 からは旧名はサポートされなくなりました。

表 C-1 は、影響を受けるキャラクタ・セットのリストです。コード内で次のキャラクタ・セットを参照するときは、新規名に置換してください。

表 C-1 廃止された NLS データ・キャラクタ・セットの新規名

旧名	新規名
AR8MSAWIN	AR8MSWIN1256
JVMS	JA16VMS
JEUC	JA16EUC
SJIS	JA16SJIS
JDBCS	JA16DBCS
KSC5601	KO16KSC5601
KDBCS	KO16DBCS
CGB2312-80	ZHS16CGB231280
CNS 11643-86	ZHT32EUC
ZHT32CNS1164386	ZHT32EUC

キャラクタ・セット CL8MSWINDOW31 は、サポートされなくなりました。新規キャラクタ・セット CL8MSWIN1251 は、実際は CL8MSWINDOW31 の複製で、以前のバージョンでは省略されていた文字をいくつか含んでいます。CL8MSWINDOW31 のかわりに CL8MSWIN1251 を使用してください。

D

用語集

用語集

1 か国語のみのサポート

1 つの言語のみのサポート。

ASCII

情報交換用米国標準コード。英語用の共通コード化 7 ビット・キャラクタ・セット。ASCII には、文字 A ～ Z と a ～ z、数字、句読点記号および制御文字が含まれる。これに対する Oracle キャラクタ・セット名は、US7ASCII。

EBCDIC

拡張 2 進化 10 進コード。IBM システムで最も使用されるコード化キャラクタ・セット・ファミリ。

EUC

拡張 UNIX コード。アジアの UNIX システムで使用される共通のコード化方法。1 つのデータ・ストリームには、異なるコード化キャラクタ・セットを 4 つまで結合できる。

ISO

国際標準化機構。

ISO 8859

8 ビット・コード化キャラクタ・セット・ファミリ。最も一般的なものは、ISO 8859-1（Latin-1 として知られている）で、西ヨーロッパ諸国で使用されている。

ISO/IEC 10646

現在、世界で使用されているほとんどの主要文字を定義しているユニバーサル・キャラクタ・セットの規格。1993 年には、ISO によって Unicode バージョン 1.1 が ISO / IEC 10646-1:1993 として承認されている。ISO / IEC 10646 には、2 バイト固定幅形式の UCS2 と 4 バイト固定幅形式の UCS4 の 2 つの形式がある。実際には 3 つのレベルがあり、すべてのレベルは複合文字のサポートに関係する。レベル 1 では、複合文字をサポートする必要はない。レベル 2 では、特定の文字（アラビア文字、タイ文字などのほとんどの Unicode 文字を含む）をサポートする必要がある。レベル 3 では、あらゆる言語において制限された複合文字をサポートする必要がある。

ISO 通貨

各国通貨を示すために使用される 3 文字の略称で、ISO 4217 規格に基づいている。たとえば、「USD」はアメリカ合衆国のドルを表す。

Latin-1

正式には、ISO 8859-1 キャラクタ・セット規格として知られている。ASCII に対する 8 ビット拡張機能で、西ヨーロッパで最も頻繁に使用される共通のラテン文字を含む 128 文字が追加されている。これに対する Oracle キャラクタ・セット名は、WE8ISO8859P1。詳細は、「ISO 8859」を参照。

NCHAR キャラクタ・セット

NCHAR、NVARCHAR2 および NCLOB 列に指定できるデータベース・キャラクタ・セットの代替キャラクタ・セット。NCHAR キャラクタ・セットは、データベース・キャラクタ・セットとは異なり、固定幅マルチバイト・キャラクタ・セットをサポートしている。NCHAR キャラクタ・セットの文字レパートリはデータベース・キャラクタ・セットにも含まれていなければならないので、NCHAR キャラクタ・セットを選択する場合は十分に注意する必要がある。

Net8

Net8 を使用すると、複数のコンピュータで Oracle Server を実行し、サード・パーティーのネットワークを介してデータを交換することができる。通信プロトコルには依存しない。

NLS

各国語サポート。NLS によって、ユーザーは母国語でデータベースと対話できる。さらに、アプリケーションをさまざまな言語および文化の環境で実行できる。

NLSDATA

接尾辞 .nlb の付いた多くのファイルの内容を指す一般的な用語。これらのファイルには、NLSRTL ライブラリが特定の NLS サポートを提供するために使用するデータが含まれている。

NLSRTL

各国語サポート・ランタイム・ライブラリ。このライブラリは、ロケールに依存しない国際化に関するアルゴリズムを提供する。ロケール固有情報（たとえば、NLSDATA）は、実行中に NLSRTL によって読み込まれる。

SQL*Net

現在は、Net8 と呼ばれている。Net8 を使用すると、複数のコンピュータで Oracle Server を実行し、サード・パーティーのネットワークを介してデータを交換することができる。通信プロトコルには依存しない。

UCS-2

UCS とは、ユニバーサル・コード化キャラクタ・セットの略語。1993 ISO および IEC 規格のキャラクタ・セット。

UCS2

固定幅の 16 ビット Unicode。それぞれの文字に、16 ビットの領域が必要。Latin-1 文字は、この規格の最初の 256 コード・ポイントにあたる。そのため、Latin-1 の 16 ビット拡張とも言える。Oracle の NLS ランタイム・ライブラリでは、このキャラクタ・セットはまだサポートされていない。

UCS4

固定幅の 32 ビット Unicode。それぞれの文字に、32 ビットの領域が必要。UCS2 文字は、この規格の最初の 65,536 コード・ポイントにあたる。そのため、UCS2 の 32 ビット拡張とも言える。これは、ISO-10646 とも呼ばれる。ISO-10646 規格は、32768 段階で最高 2,147,483,648 文字まで指定でき、これの最初の段階が UCS2 セットである。ISO 規格は異なるコード間での変換も指定する。

UCS および UTF 形式間の Unicode マップ

次に、Unicode 関連のキャラクタ・セットが文字コード値の範囲に関してどのように相互に参照しているかを示す。

UCS2	UTF8	説明
0x0000 ～ 0x007F	0x00 ～ 0x7F	シングル・バイト
0x0080 ～ 0x07FF	0xC0 ～ 0xDF	2 バイトのシーケンス・リーダー（5+6 ビット）
0x0800 ～ 0xFFFF	0xE0 ～ 0xEF	3 バイトのシーケンス・リーダー（4+6+6 ビット）
	0x80 ～ 0xBF	後続のバイト（それぞれ 6 ビット）

UCS4	UTF8	説明
0x00000000 ～ 0x0000007F	0x00 ～ 0x7F	シングル・バイト
0x00000080 ～ 0x000007FF	0xC0 ～ 0xDF	2 バイトのシーケンス・リーダー（5+6 ビット）
0x00000800 ～ 0x0000FFFF	0xE0 ～ 0xEF	3 バイトのシーケンス・リーダー（4+6+6 ビット）
0x00001000 ～ 0x001FFFFF	0xF0 ～ 0xF7	4 バイトのシーケンス・リーダー（3+6+6+6 ビット）
0x00200000 ～ 0x03FFFFFF	0xF8 ～ 0xFB	5 バイトのシーケンス・リーダー（2+6+6+6+6 ビット）
0x04000000 ～ 0x7FFFFFFF	0xFC ～ 0xFD	6 バイトのシーケンス・リーダー（1+6+6+6+6+6 ビット）
	0x80 ～ 0xBF	後続のバイト（それぞれ 6 ビット）
	0xFE ～ 0xFF	予約済みまたは未使用

UCS4	UTF16	説明
0x00000000 ～ 0x0000FFFF	0x0000 ～ 0xFFFF	UCS2 と同じ
0x00010000 ～ 0x0010FFFF	0xD800 ～ 0xDBFF	High surrogate ((x-0x10000)>>10)&0x3FF
	0xDC00 ～ 0xDFFF	Low surrogate (x-0x10000)&0x3FF
0x00110000 ～ 0x7FFFFFFF		UTF16 にはマップされていない

Unicode

Unicode はユニバーサル・キャラクタ・セットの 1 種で、16 ビットでコード化された 64K 文字の集まりのこと。大半の文字が、既存のキャラクタ・セット標準でコード化され、世界中で使用されている文字のほとんどがサポートされている。Unicode は、Unicode Inc. によって所有され定義されている。Unicode は正統なコード化方法で、その値はさまざまなロケールで使用できる。ただし、Unicode と Oracle キャラクタ・セット間でのラウンドトリップ変換が、情報を失うことなく行われるとは限らない。

Unicode のコード・ポイント

コード化テキストを処理および交換する単位を表す 16 ビットのバイナリ値。U+0000 と U+FFFF 間のすべてのポイントが、コード・ポイントである。この用語は、コード要素、コード位置およびコード値と置き換えることができる。

UTF-16

UCS4 キャラクタ・セットの拡張文字を表現するために、UCS2 のコード・ポイントの組を考慮した UCS2 の拡張要素。UCS2 には、UTF16 コード化方式をサポートする high（leading）および low（trailing）surrogates に割り当てられた範囲のコード・ポイントがある。

UTF-8

1 文字に対して 1、2 または 3 バイトの順序を使用する UCS2 の可変幅コード化方法。0 ～ 127 の文字（7 ビットの ASCII 文字）は 1 バイトでコード化され、128 ～ 2047 の文字は 2 バイト、2048 ～ 65535 までの文字は 3 バイトが必要である。これに対する Oracle キャラクタ・セット名は、UTF8（Unicode 2.0 規格の場合）。この規格では、1 文字につき 4、5 および 6 バイトの順序を使用する UCS4 文字をサポートするために、拡張用の空き領域が用意されている。

エクスポート

データをアーカイブするために、またはオペレーティング・システムと Oracle データベース間でデータを移動するために、ファイルへデータを書き込むこと。

絵文字

表示デバイスまたは用紙でのグラフィカルな文字の表現。たとえば、H、H または H はそれぞれ異なる絵文字であるが、同じ文字を表す。

各国通貨

ある国または地域で使用されている通貨記号。たとえば、「$」はアメリカ合衆国のドルを表す。

キャラクタ・セット変換

あるコード化キャラクタ・セットの別のキャラクタ・セットへの変換。

クライアント・キャラクタ・セット

クライアントが使用するコード化キャラクタ・セット。クライアント・キャラクタ・セットは、データベース・サーバーのキャラクタ・セットとは異なる場合がある。この場合、キャラクタ・セット変換を行う必要がある。

結合文字

先行する基本の文字とグラフィカルに結合する文字。これらの文字は、単独では使用されない。アクセント記号、特殊記号、ヘブライ・ニクダ、アラビア発音記号、インド・マトラスなどの文字が含まれる。

言語ソート

文字列のバイナリ表現ではなく、ロケールに関する要件に基づいた文字列のソート。

言語索引

言語上の照合順序に基づいた索引。

コード化キャラクタ・セット

キャラクタ・セットのコード化とは、キャラクタ・セットと、そのキャラクタ・セットのそれぞれの文字とそのビット表現との 1 対 1 の関係を規定する一連の明確な規則である。

コード体系

詳細は、「文字コード体系」を参照。

国際化

多様な言語および文化の環境において使用できるように、ソフトウェアを柔軟に作成するプロセス。国際化と、ローカライゼーションを混同してはいけない。ローカライゼーションとは、ソフトウェアをある固有のロケールで使用できるように準備するプロセスのこと。

サーバー・キャラクタ・セット

データベース・サーバーで使用されるキャラクタ・セット。

照合

すべての文字列の順序付け（アルファベットから線形順序まで）。照合は、言語ソートまたはバイナリ・ソートの順序付けで使用される。

制限なし多言語サポート

希望するだけの多数の言語を使用できる。Unicode などのユニバーサル・キャラクタ・セットを使用すると、無制限に多言語をサポートできるようになる。これは、ユニバーサル・キャラクタ・セットが非常に大きい文字レパートリをサポートしているためで、この文字レパートリには世界のほとんどの現代語が含まれている。

制限付き多言語サポート

関連する言語のグループに制限された多言語サポート。すべての言語ではなく、関連する言語をサポートする。西ヨーロッパ諸国の言語など、類似する言語ファミリが ISO 8859/1 などで表現される。ただし、この場合はタイ語は含まれない。

置換文字

希望する文字がターゲット・キャラクタ・セットで使用可能でなかった場合、その文字の変換中に使用される文字。たとえば、多くの場合には「?」が Oracle のデフォルトの置換文字として使用される。

データベース・キャラクタ・セット

コード化キャラクタ・セット。

バイナリ・ソート

バイナリのコード表現に基づいた文字列のソート。

発音区別記号

通常、発音またはストレスに関する情報を提供する、文字に付けられたマーク。

表意文字

概念を表現する記号。中国語は、表意的記述法の例。

フォント

キャラクタ・セット内の文字をグラフィカルに表現する順序付けられた絵文字の集まり。

複合文字

複合文字順序によって表現される 1 文字。このタイプの文字は、ヨーロッパ地域の言語で使用される多くのラテン文字と同様に、タイ語、ラオ語、ベトナム語および韓国語の文字に見られる。

複合文字順序

基本の文字の後に 1 つ以上の結合文字が続いてできる文字。結合文字とも呼ばれる。

マルチバイト文字

1 つ以上のバイトで表現されるコード化文字。マルチバイトのデータ・ストリームには可変幅の文字を入れることができるので、個々の文字のテキスト処理を大量に行うことができる。詳細は、「ワイド・キャラクタ」を参照。

文字

文字、発音区別記号の付いた文字、数字、象形文字、句読点、記号などのデータを表すために使用する独立した単位。

文字コード化体系

コード化キャラクタ・セットを定義するときに使用するマップのタイプ。Oracle では、シングルバイト、マルチバイト、シフト・センシティブ・マルチバイトおよび固定幅のキャラクタ・セット・コードを含む、たくさんのキャラクタ・セット・コードがサポートされている。

文字の分類

文字の分類によって、それぞれの有効なキャラクタ・コードに対応付けられた文字タイプに関する詳細情報が提供される。つまり、その文字がアルファベットであるか、大文字であるか、小文字であるか、句読点であるか、制御文字であるか、またはスペース文字であるかどうかが文字の分類情報として提供される。

文字変換

文字の大文字から小文字、または小文字から大文字への変換。

ユーロ

欧州連合の参加メンバーである州が使用する新しい通貨貨幣。

ローカライゼーション

言語固有または文化固有の情報をソフトウェア・システムに提供するプロセス。アプリケーションのユーザー・インタフェースの翻訳は、ローカライゼーションの 1 つの例である。ローカライゼーションと、国際化を混同してはいけない。国際化とは、多様な言語および文化の規則を扱えるようにソフトウェアを一般化するプロセスのこと。

ロケール

特定の地域で参照される言語的および文化的な情報の集まり。一般的に、ロケールは NLS データ・ファイルに定義されている言語地域、キャラクタ・セット、言語処理およびカレンダ情報で構成される。

ワイド・キャラクタ

固定幅の文字形式。固定幅の領域でデータが処理されるので、大量のテキスト処理に適している。ワイド・キャラクタは、内部の文字処理をサポートすることを目的としている。したがって、インプリメンテーションに依存する。

索引

O

R

T

U

V

W

あ

う

お

か

き

く

け

こ

さ

し

す

せ

そ

ゆ

よ

り

れ

ろ