4D for Oracle

リファレンスガイド

Windows® and Mac™OS版

4D for Oracle
by
Olivier Boulland

4D for Oracle リファレンス
Mac™ OS and Windows® Versions 1.5

注意

このソフトウェアの使用に際し、本製品に同梱のLicense Agreement（使用許諾契約書）に同意する必要があります。ソフトウェアを使用する前に、License Agreementを注意深くお読みください。

このマニュアルに記載されている事項は、将来予告なしに変更されることがあり、いかなる変更に関してもACI SAおよびACI USは一切の責任を負いかねます。このマニュアルで説明されるソフトウェアは、本製品に同梱のLicense Agreement（使用許諾契約書）のもとでのみ使用することができます。
ソフトウェアおよびマニュアルの一部または全部を、ライセンス保持者がこの契約条件を許諾した上での個人使用目的以外に、いかなる目的であれ、電子的、機械的、またどのような形であっても、無断で複製、配布することはできません。

Author: Loïc Vandereyken

目次

4D for Oracleは、4th Dimensionのデータベースと Oracleのデータベース間のやり取りを行うことができる4th Dimensionの外部ルーチンです。4D for Oracleを使用することにより、4th Dimensionのデータベース上でOracleデータベースに格納されているデータを表示したり、または操作、修正することができます。

このマニュアルについて

このマニュアルは、4th Dimensionのデータベースと Oracleサーバを統合したシステムの実行方法や使用方法、および修正方法について説明しています。

また、このマニュアルはすでに4th Dimensionのプログラム言語および OracleのSQL言語に精通しているユーザを対象に記述されています。そのため、4th Dimensionまたは Oracleを初めて触れる方は、まずこの両製品に馴れ親しんでからこのマニュアルをお読みになることをお勧めします。

このマニュアルは、次の3つの部分から構成されています。

■ 第1章の「はじめに」では、4D for Oracleの主な概要について紹介し、ユーザがフロントエンドアプリケーションを開発する際に必要となるいくつかの設計選択について説明します。

■ 第2、3、4章では、2つのデータ管理方法について説明します。1つは、コンテキストを使った4th Dimensionのデータと Oracleのデータのリンク方法。もう1つは、ローレベルコマンドを使った Oracle サーバ上でのSQL文の送信および実行方法。

■ 第5章から第9章までは4D for Oracleの「ランゲージリファレンス」です。これらの章では、4D for Oracle言語の各コマンドについて説明します。

尚、次ページの各章の説明の中でこのマニュアルの構成について詳しく説明します。

クロスプラットフォームマニュアルの取り扱い方法

このマニュアルは、Macintosh と Windows 両方の環境における使用方法を説明します。2つのプラットフォーム上で 4D Backup の考え方や機能はほとんど同じですが、必要がある場合は、その違いについても説明があります。こうした違いには、表示上のユーザインタフェースやキーボードコマンドも含まれます。

このマニュアルにはMacintosh、Windows 両方の環境の図を示してありますが、主にMacintosh版の図を中心に構成されています。Windows版の4D for Oracleをお使いの方は、あらかじめご了承ください。

4th Dimensionと4D for Oracle

4D for Oracleは、4th Dimensionまたは4D Serverで使用することができます。4D for Oracleと一緒に使用すると、4th DimensionはOracleデータベースの任意のクライアントになれるデータベースを作成することができます。そのデータベースのコピーで各ユーザは同時にOracleデータベースに接続し、使用することができます。

4D Serverは、マルチユーザ用のデータベースアプリケーションを作成することができます。4D for Oracleと一緒に使用すると、4D Serverは複数のユーザがOracleデータベースに接続できるようにします。

このマニュアルでは、4th Dimensionと4D Serverはこの両製品の機能における違いがある場合を除いて4th Dimensionとして説明されています。

マニュアルの構成

このマニュアルは、以下のような構成になっています：

- 第1章：「はじめに」では、4D for Oracle の概要を紹介します。
- 第2章：「設計選択」では、Oracle RDBMSのフロントエンドとして4th Dimensionを使用する際のいくつかのインプリメントを説明します。
- 第3章：「コンテキストの使用」では、データやり取りにおける管理方法について説明します。
- 第4章：「ローレベルコマンドの使用」では、4th DimensionとOracle間のやり取りを管理するコマンドの使用方法について説明します。
- 第5章：「ログインコマンド」では、Oracleサーバからのログインとログアウト、および接続中のハイレベル操作を実行することができるコマンドについて説明します。
- 第6章：「コンテキストコマンド」では、4th Dimension内でのOracleデータのロードと更新を実行できるコマンドについて説明します。
- 第7章：「ローレベルコマンド」では、4th DimensionとOracle間のやり取りを管理するコマンドについて説明します。
- 第8章：「環境設定コマンド」では、コンテキスト、カーソルおよび接続における環境設定を行うコマンドについて説明します。
- 第9章：「制御コマンド」では、4th Dimension内でのデータロードの取り消しおよび発生したエラーを管理することができるコマンドについて説明します。
- 付録A：「タイプ変換」では、Oracleと4th Dimension間でのタイプ変換のプロセスを要約してします。
- 付録B：「4D for Oracleの関数によって返される値」では、4D for Oracleの関数によって返される値を一覧表示しています。
- 付録C：「エラーコード」では、4D for Oracleの中で起こったエラーコードを一覧表示しています。

表記方法について

4D Backupのマニュアルでは、内容が判りやすいように特定の規則を設けています。処理手順はボールド体で記述され、通常、短いコメント・解説が付いています。処理手順や解説の表記フォーマットは次のようになっています。

1. 番号の付いた文章は、何をすべきかを指示する。
指示の後に解説・コメントが付きます。

番号付きの文章で入力データを指示する場合は、入力する文字を引用符で囲んで次のように記述します。

2. 名字フィールドに、" 吉野 " と入力する。
空白や句読点も含め、引用符で囲まれた文字は正確に入力します。

次のような注記もあります。

注: このマークは、プログラムの特定機能について、操作上のコメントを記述する場合に使いますが、要点だけを知りたい場合はこれにとらわれず先に進んでください。

このような注意書きは、重要な情報に対する注意を促しています。

このような警告は、データが失われる恐れがあることを示します。

4D for Oracleコマンドには ODという頭文字を付けて、4th Dimensionのコマンドや別のモジュールによって追加されたコマンドと区別しています。

4D for Oracleのコマンドは、すべて大文字で表されます。例えば、

OD BIND TOWARDS SQL

4D for Oracleの関数は、ODの後の最初の1文字だけ大文字で表されます。例えば、

OD Number of columns

注：4D for Oracleのコマンドと関数は、本文中および例題プロシージャの中では、ボールドのイタリック体で表記されています。例えば、

OD BIND TOWARDS SQL*、*OD Number of columns

また、テーブル名はフィールド名、フォーム名、および他の項目名と区別するために本文中では角括弧で囲まれています。例えば、会社テーブルは、" [会社] " と表されます。

この章では、4D for Oracleの下記のような基本的な事柄について説明します。

■ 4D for Oracleの機能

■ 4D for OracleとOracleの統合

■ 4D for Oracle使用によるデータ管理

■ 4th Dimensionでの4D for Oracleの使用

概要

4D for Oracleは、4th Dimensionのデータベースと Oracleのデータベース間のやり取りを行うことができる4th Dimensionの外部ルーチンです。4D for Oracleを使用することにより、4th Dimensionのデータベース上でOracleデータベースに格納されているデータを表示したり、または操作、修正することができます。

4D for Oracleを使うと、4th DimensionはOracleデータベースのフロントエンドになります。このシステム構成により、次のようなことを行うことができます：

■ カスタムメニューやカスタムフォームを備えた、完全にカスタマイズされた4th Dimensionインタフェースの表示

■ Oracleのデータ保存およびディスクアクセス機能を使用した強力なデータ管理システムの作成

■ 同一のOracleデータベースにアクセスする4th Dimensionのマルチデータベースの作成

4th DimensionとOracle

4D for Oracleを使用することにより、4th DimensionとOracleの最適な組み合わせを持ったデータ管理システムを作成することができます。また、4D for Oracleを使うと、4th DimensionとOracleはタスクが分散され、ネットワーク負荷が大幅に軽減されたクライアント/サーバ構造において、お互いがインテリジェントに機能しあいます。

4th Dimensionのインタフェース

4th Dimensionは、直観的でグラフィカルなフォーム、エディタ、メニュー等を使ってデータを管理できる簡単でしかも操作性に優れたインタフェースを持っています。

4th DimensionとOracle間のやり取りは、ユーザに対してとてもわかりやすいように設計されます。ユーザはレコードの作成や修正、およびOracleテーブルへのアクセス、そのテーブルの行(rows)への更新が行われていることを意識することなく、データを処理できます。

4th Dimensionのインタフェースを使えば、ユーザは4th Dimensionのプログラム言語やOracleのSQL言語を知らなくてもデータベース処理を行うことができます。

Oracleのデータ管理能力

Oracleは膨大な量のデータを管理したり、ユーザに送信されるデータを即座に取り出すことができます。Oracleは、一般的にパーソナルコンピュータ(パソコン)が持っているデータ保存やディスクアクセス機能をはるかに上回る処理能力を持っているミニコンピュータ(ミニコン)やメインフレーム上で操作します。

最適化されたクライアント/サーバアーキテクチャ

4D for Oracleのクライアント/サーバアーキテクチャでは、データは必要な場合にのみネットワーク上を循環するのでネットワーク負荷を軽減させることができます。そのネットワークは、単に任意のファイルやメッセージを運ぶだけの転送ツールに留まりません。4D for Oracleを使うことにより、そのネットワークは通信ツールにもなります。

クライアント/サーバアーキテクチャでは、タスクはクライアントとサーバの間で分散されます。各マシンは、最適化されたタスクを実行します。

■ クライアントはネットワークを通してサーバに接続され、共通のデータにアクセスします。4D for Oracleは問い合わせ(クエリー)を生成して、ローカルデータを処理し、その処理を実行するためにサーバを呼び出します。

■ サーバは共通のデータを保存します。そして、結果(リザルト)を得るために必要な処理を実行してクライアントの問い合わせに応答します。

分散タスク

4D for Oracleを使うと、エンドユーザマシンは単にホストとやり取りができる端末以上の働きをします。マシンはデータのローカル処理を管理し、Oracleサーバを各ユーザの問い合わせにおけるインタフェースの制御から解放します。分散タスクは、多くのサーバ機能を正常に使いこなします。

4D for Oracleを使ったデータ管理

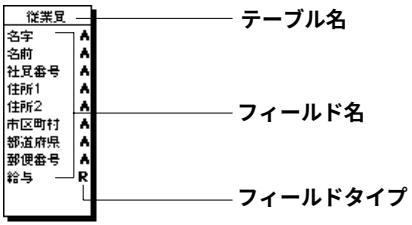
代表的なネットワーク接続では、ユーザは4th Dimensionのデータベースからサーバに接続します。そして、ユーザはサーバが配置または返すデータに問い合わせします。例えば、ユーザは年収5,000,000円以上の従業員のすべてに問い合わせするとします。

ここでのポイントは、ユーザが4th Dimensionを使ってそのデータを操作したり、あるいは追加、修正できる点です。例えば、ユーザが従業員の住所を修正したり、あるいは新規に従業員を追加することができます。データが追加または修正されと、そのデータはOracleデータベース内で更新されなければなりません。

4th DimensionとOracleのデータストラクチャ

4th DimensionおよびOracleは、異なるデータストラクチャを使ってデータを格納します。つまり、4th Dimensionはテーブルとフィールド、Oracleはテーブル(table)とカラム(columns)の中でデータを組織化します。

例えば、次の図は4th Dimensionのストラクチャファイルを示しています。



このストラクチャは、次のOracleのテーブル定義と同じ意味を持っています。

Column Name	Type
First Name	Char(15)
Last Name	Char(15)
EmployeeNum	Char(2)
Address1	Char(30)
Address2	Char(30)
City	Char(20)
State	Char(8)
Zip	Char(10)
Salary	Number

データタイプ変換

4th Dimensionの各フィールドは特定のデータタイプを持ちます。一方、Oracleの各カラムは異なるデータタイプを持ちます。そのため、4th DimensionとOracleの間でデータ変換を行うと、4D for Oracleは異なるデータタイプ間の必要な変換を実行します。データタイプの変換に関する詳細は、付録Aを参照してください。

NULL値について

カラムは異なるデータタイプを持っている他に、NULL値またはNULLでない値のどちらかを持つことができます。もし、カラムの値がNULLでない場合は、行を更新する前にカラムに値を入力しておく必要があります。

もし、カラムの値がNULLの場合は、そのカラムに値を入力しなくても構いません。例えば、次のOracleテーブルでは、「Address2」カラムは必須入力ではないので、2行目の住所にデータが入力されていない場合があります。

Column Name	NULL?	Type
First Name	Not NULL	Char(15)
Last Name	Not NULL	Char(15)
EmployeeNum	Not NULL	Char(8)
Address1	Not NULL	Char(30)
Address2	NULL	Char(30)
City	Not NULL	Char(20)

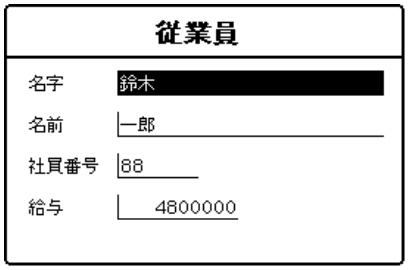
Column Name	NULL?	Type
State	Not NULL	Char(2)
Zip	Not NULL	Char(15)
Salary	Not NULL	Number

ある行の「Address2」カラムがNULL値を含んでいれば、そのNULL値が4th Dimension内でどのように表示されるのかを指定することができます。例えば、従業員の2行目の住所欄に何も入力されていないことを示すには、4th Dimensionの当該フィールドに“ N/A ”を表示します。言い換えると、“ N/A ”が4th Dimensionのフィールドに入力されていれば、Oracleの行をNULL値のままで更新することができます。

4D for Oracleの***OD SET NULL VALUE***コマンドを使用することにより、各データタイプと同等のNULL値を指定することができます。このマニュアルで記述されている例題では、文字フィールド「住所2」と同等のNULL値は“ N/A ”です。もし、同等のNULL値を指定しない場合は、4th Dimensionはそのデータタイプと同等のデフォルトNULL値を使用します。

4th DimensionのレコードとOracleの行

4th Dimensionはレコードにデータを格納します。各レコードはテーブルに属し、そのテーブル内に各フィールドの入力エリアを持つことができます。例えば、次のレコードは[従業員]テーブル内にそのフィールドの入力エリアを持っています。



Oracleは行にデータを格納します。各行はテーブルに属し、そのテーブル内に各カラムの入力エリアを持つことができます。例えば、次の行は「Employees」テーブル内にそのカラムの入力エリアを持っています。

First Name	Last Name	EmployeeNum	Salary
一郎	鈴木	88	4800000

Oracleを使用してデータを抽出または更新する場合、次の内容を確認する必要があります。

- 抽出または更新したい行
- 各行内の抽出または更新したいカラム
- 行が属しているテーブル

4D for Oracleコマンドを使用してデータ処理を行う場合、上記のような情報を取得する必要があります。この情報がどのように提供されるかは、ユーザがコンテキストコマンドを使用しているか、またはローレベルコマンドを使用しているかによって異なります。

コンテキストコマンドの使用

コンテキストを使用してOracleのカラムに4th Dimensionのフィールドや変数、配列をバインドすることができます。コンテキストコマンドを使用すると、SQL文を使用することなくデータを取り出し、そのデータ上で処理を行うことができます。また、4D for Oracleのコマンドを使用すると、ユーザは4th Dimensionのテーブルやフィールドを使ってステートメント(SQL文)を書くことができます。4D for Oracleがそのステートメントを解釈すると、4D for OracleはそのステートメントにふさわしいSQL文を生成します。

ローレベルコマンドの使用

ローレベルコマンドを使ってSQL文を実行することもできます。このケースでは、SQL文を使用してデータをロードし、そのデータ上で処理を行います。このSQL文において、表示または更新したい行およびカラムを指定します。

プライマリ(主)キー

Oracleでは、プライマリキーはテーブル内の行に重複したものがないかどうかを調べるために使用されるカラムまたはカラムの連結したもので構成されます。

例えば、顧客が一覧表示された次のようなテーブルがあったと仮定します。

First Name	Last Name	Address
研二	山下	新宿東3-35-20
陽子	下川	渋谷西5-15-12
智	鈴木	銀座南7-20-10
誠治	鈴木	池袋北9-1-25

この場合、複数の顧客が同じ名字を持つ可能性があるので「Last Name」カラムをプライマリキーにしても意味がありません。実際、上記の例では“ 鈴木 ”という名字の顧客が2人います。

そのため、複数のカラムを使用してプライマリキーを定義する必要があります。2人の顧客が同じ名前でしかも同じ住所であることはまず考えられないので、「Last Name」と「Address」を連結したものをプライマリキーに定義することができます。上記のテーブルでは“ 鈴木 ”という名字の顧客が2人存在しますが、“ 池袋北9-1-25 ”に住んでいる鈴木さんは1人しかいません。

4th Dimension側では、データベースのストラクチャは次のような規則に注意して設計されている必要があります。

■ プライマリキーが1つのカラムのみで構成されている場合は、そのプライマリキーに対応するフィールドはインデックス付きでしかも重複不可の属性を持っていなければならない。

■ プライマリキーが複数のカラムから構成されている場合は、その目的用に作成された文字フィールドと対応しているフィールドをリレートする必要があります。そのリレートフィールドは、インデックス付きでしかも重複不可の属性を持っていなければならない。

4th Dimensionのフィールドや変数、配列とOracleのカラムをリンクするためにコンテキストを定義する場合、プライマリキーが何であるかを知っておく必要があります。そして、コンテキストの定義に同じプライマリキーを指定する必要があります。上記の例で説明すると、コンテキストの定義はプライマリキーの2つのコンポーネント(要素)である「Last Name」と「Address」を含んでいる必要があります。

4th Dimension使用による4D for Oracle

4D for Oracleは、4th Dimensionまたは4D Serverで使用することができます。4D for Oracleと一緒に使用すると、4th DimensionはOracleデータベースの任意のクライアントになれるデータベースを作成することができます。そのデータベースのコピーで各ユーザは同時にOracleデータベースに接続し、使用することができます。

4D Serverは、マルチユーザ用のデータベースアプリケーションを作成することができます。4D for Oracleと一緒に使用すると、4D Serverは複数のユーザがOracleデータベースに接続できるようにします。この場合、クライアントは4D Serverが動作しているMacintoshサーバに接続したままになっているにも関わらず、そのクライアントは直接Oracleサーバとやり取りを行い、データを表示したり修正することができます。

このマニュアルでは、4th Dimensionと4D Serverはこの両製品の機能における違いがある場合を除いて4th Dimensionとして説明されています。

初期設定(4D for Oracle Preferences)ファイルの格納場所

(Macintosh上のシステムフォルダの「初期設定」フォルダ内、またはWindows下の「Windows」ディレクトリ内に作成される）特別な“ ACI ”フォルダにすべてのACI社製品の初期設定ファイルが格納されます。4th Dimensionまたは4D Clientを使用している場合、4D for OracleはこのACIフォルダ内の該当する初期設定ファイルを探しに行きます。この初期設定ファイルの名前は、MacOS下では「4D for Oracle preferences」、Windows下では「4DORACLE.PRF」と命名されています。

注：4D for Oracle Preferencesファイルは、1番最後に接続した際のログイン名やパスワード、デバッグウインドウの位置などの情報を格納しています。しかし、もし4D for Oracleがこのファイルを見つけることができない場合は、デフォルトの値で新しく4D for Oracle Preferencesファイルを作成します。

OCIの使用とSQL*Netのサポートについて

4D for Oracleは、Oracleの機能をフルに活用できるOracle Call Interfaces(通称、OCI)を使用します。これにより、4D for Oracleは快適なパフォーマンスを得ることができます。

4D for Oracleは、Oracle社から提供されているOCIドライバファイルを必要とします。このファイルはSQL*Netだけでなく Oracleと一緒に機能します。

ユーザは、Oracleリレーショナルデータベース管理システム (RDBMS)のフロントエンドとして4th Dimensionを使用することができます。このように、ユーザは4th Dimensionの簡単でしかも使いやすいインタフェースを使用することによりOracleのデータ記憶機能およびデータ操作機能を利用することができます。4th Dimensionは、いろいろなユーザインタフェースを作成することができます。

このシステム構成において、4D for Oracle使用による効果的な部分はそのシステムがクライアント／サーバアーキテクチャを基にしているところです。4D for Oracleを使用することにより、クライアントの4th Dimensionアプリケーションは、どのデータ処理機能が実行されるのかをOracleサーバに知らせます。すると、サーバはその機能を実行し任意の必要なデータを返します。クライアント／サーバを基にしたアプリケーションでは、データは常にサーバ上に置かれています。

インプリメントの選択

ユーザは4th Dimensionアプリケーションを作成すると、おそらくデータベース設計におけるクライアント／サーバアーキテクチャの効果的な部分を利用したくなるでしょう。データベースを設計するための選択メソッドは、データの転送速度に重要な影響を及ぼします。

次の節では、4D for Oracleを使用したデータベースにおける2つのインプリメントメソッドについて説明します。

コンテキストを使ったインプリメント

コンテキストを使用する場合、4th Dimensionのフィールドや変数とOracleサーバ上のカラムとの間のリレート関係を記述します。例えば、4th Dimensionの「vFirstName」変数とOracleテーブルの「FirstName」を結び付けることができます。コンテキストを一度作成すれば、、4th Dimensionのフィールドや変数、配列によってデータを操作するための4D for Oracleコマンドを使用することができます。

コンテキストは、迅速かつ簡単にデータをOracleデータベースから取り出し、クライアントの4th Dimensionアプリケーションの中で表示および修正した後、Oracleサーバに返すことができます。コンテキストは、Oracleデータベースのフロントエンドとして実行している既存の4th Dimensionデータベース取得用に最も利用されます。データベースをさらに効果的にするには、そのデータベースの中にローレベルコマンドを組み込んだ方法に移行する必要があります。

もし、データベースのインプリメントにコンテキスト使用を選択した場合は、パフォーマンスに関する重要事項を記述した次の節を読む必要があります。

4th Dimensionフィールドを使用したバインド

一般的に、4th DimensionのフィールドとOracleのカラムのバインド作成は避けたほうが賢明です。4th Dimensionのフィールドを含んだバインド作成は、Oracleデータベースストラクチャを反映する4th Dimensionデータベースストラクチャを保持する必要があります。

しかし、バインドを使用するためにデータベースストラクチャの複製を作成する必要はありません。コンテキストは、変数や配列を使って設定することができます。これらは、クライアント／サーバアーキテクチャにより効果的にアプローチできます。

フィールドを使ったバインド使用の有効な手段の1つにバッチ処理があります。バッチ処理をインプリメントするには、取り出したデータをローカルに格納するためにフィールドを使ったバインドが使用されます。そして、いったんOracleサーバとの接続を終了し、クライアントマシン上でそのデータを使って作業します。その後、修正したデータを戻すために再度Oracleサーバに接続します。

4th Dimension変数または配列を使用したバインド

バインドに4th Dimensionの変数または配列を使用すると、ホストデータベースを複製する4th Dimensionデータベースの保持を回避することができます。ユーザのバインドは4th Dimensionのフィールドを使用しないので、データベースは1つのテーブルとフィールドだけで十分です。データを転送し操作するには、任意のフォーマットで変数または配列を表示するフォームを作成します。

▼ 次の例は、コンテキストが変数を使ってどのように定義されるのかを示しています。各変数はまずタイプ宣言され、次にホストデータベース内のカラムと結び付けられています。

```
C_INTEGER (vCust_num ; vCust_rep)
C_STRING (50 ; vCust_name ; vCust_city)
C_STRING (255 ; vCust_addr1 ; vCust_addr2)
C_STRING (8 ; vCust_state)
C_STRING (10 ; vCust_zip)
C_REAL (vCredit ; vReceivable)

context := OD Create context ("SCOTT.CUSTOMERS")
If (context # 0)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_NUM" ; >>vCust_num ; 1)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_NAME" ; >>vCust_name ; 1)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_REP" ; >>vCust_rep ; 0)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_ADDR1" ; >>vCust_addr1 ; 0)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_ADDR2" ; >>vCust_addr2 ; 0)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_CITY" ; >>vCust_city ; 0)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_STATE" ; >>vCust_state ; 0)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_ZIP" ; >>vCust_zip ; 0)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_CREDIT" ; >>vCust_credit ; 0)
   OD ADD TO CONTEXT (context ; "SCOTT.CUSTOMERS.CUST_RECEIVABLE" ;
   >>vCust_receivable ; 0)
End if
```

ローレベルコマンドを使ったインプリメント

ローレベルコマンドを使ってデータベースを設計する際、実行するデータ処理をOracleサーバに知らせるためにSQL文を使用します。ユーザは、あるタイプのデータ処理を実行するためのSQL文を作成することができます。また、サーバ上で行（rows）を選択、追加、更新、削除することができます。データのセレクションは、一定の条件文またはリレートデータを含んだ条件文を基に作成されます。

一般的に、ローレベルコマンドを使った方がコンテキストを使った場合よりも効果的です。それは、コンテキストコマンドでは、4D for Oracleはサーバ上でリクエスト（要求）を実行する前にそのコマンドをSQLクエリー（問い合わせ）に翻訳しなければなりません。これに対して、ローレベルコマンドでは、リクエストはすでにSQL用語で設定されています。4D for Oracleは、ローレベルコマンドをOracleが理解できるSQL文（ステートメント）に翻訳するためにOCIs (Oracle Call Interfaces)を使用します。また、ローレベルコマンドでは、カーソル内部でSQL文を実行します。カーソルとは、SQL文を参照または制御するために使用されるメモリバッファのことです。各カーソルは、単一のSQLクエリーを制御します。複数のカーソルを作成し、同一接続内で同時に複数のステートメントを使用することができます。

変数または配列を使ったコンテキストと同じように、ローレベルコマンドもデータベースストラクチャを保持する必要がないので4th Dimensionのフィールドを必要としません。あるフォーマットでデータを表示するフォームを作成するために1つのテーブルとフィールドがあるだけで十分です。

▼ 次の例は、ホストデータベース内の一連の行を選択し先頭行のデータを取り出すためにローレベルコマンドがどのように使用されているかを示しています。メソッドはまず選択ステートメントを送信し実行します。次に、1カラムづつ先頭行のデータをロードします：

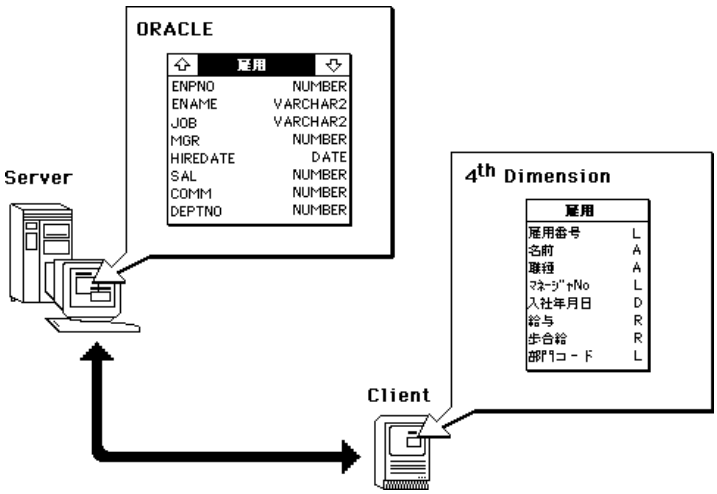
```
`フォーム上でのデータ表示用に使用される変数の宣言
C_INTEGER (vCust_num ; vCust_rep)
C_STRING (50 ; vCust_name ; vCust_city)
C_STRING (255 ; vCust_addr1 ; vCust_addr2)
C_STRING (8 ; vCust_state)
C_STRING (10 ; vCust_zip)
C_REAL (vCredit ; vReceivable)
C_TEXT (vSQL)

`SQLクエリーの作成
vSQL := "SELECT CUST_NUM, CUST_NAME, CUST_REP, CUST_ADDR1, CUST_ADDR2,
CUST_CITY,"
vSQL := vSQL + "CUST_STATE, CUST_ZIP, CREDIT, RECEIVABLE FROM CUSTOMERS"
$result := OD Set SQL in cursor (ID_Cursor ; vSQL)
OD BIND TOWARDS 4D (ID_Cursor ; 1 ; >>vCust_num)
OD BIND TOWARDS 4D (ID_Cursor ; 2 ; >>vCust_name)
OD BIND TOWARDS 4D (ID_Cursor ; 3 ; >>vCust_rep)
OD BIND TOWARDS 4D (ID_Cursor ; 4 ; >>vCust_addr1)
OD BIND TOWARDS 4D (ID_Cursor ; 5 ; >>vCust_addr2)
OD BIND TOWARDS 4D (ID_Cursor ; 6 ; >>vCust_city)
OD BIND TOWARDS 4D (ID_Cursor ; 7 ; >>vCust_state)
OD BIND TOWARDS 4D (ID_Cursor ; 8 ; >>vCust_zip)
OD BIND TOWARDS 4D (ID_Cursor ; 9 ; >>vCredit)
OD BIND TOWARDS 4D (ID_Cursor ; 10 ; >>vReceivable)
OD EXECUTE CURSOR (ID_Cursor)
n:= OD Load rows cursor (ID_Cursor ; 1)
```

データのやり取りを管理するために4D for Oracleを使用する最も簡単な方法は、コンテキストを作成することです。「コンテキスト」は、Oracle内のデータと4th Dimension内のデータの関係付け(リレート)を定義したコードが記述されたもので、Oracleサーバ上で選択されたデータを指定します。

例えば、4th Dimensionの[従業員]テーブルがOracleのEMPテーブルと同じ項目を持っていると仮定します。この場合、両者間をデータが転送できるように4th Dimensionのテーブル([従業員])とOracleのテーブル(EMP)を接続するために、[従業員]テーブルの各フィールドとEMPテーブルのカラムをバインドするコンテキストを作成します。

「バインド」は、両者のリレートオブジェクト間をリンクするものとして考えられたものです。

次の図は、コンテキストによってリンクされた4th DimensionのテーブルとOracleのテーブルを示したものです。EMPテーブルの各カラムはバインドを通して、[従業員]テーブルのフィールドとリンクされています。



コンテキストが実行されると、4D for Oracleはサーバ上の一連の行を選択するSELECT文を生成します。もし、コンテキストに追加されるSQL句が何もない場合は、コンテキストの実行によりEMPテーブル内のすべての行が選択されます。

そして、これらの行は4th Dimension内にロードされ、操作されます。

コンテキストコマンドについて

ここでは、データ管理を行うために使用されるコンテキストコマンドの一連操作の流れについて説明します。

1. ***OD Login*関数または*OD Login dialog*関数を使って接続を作成する。**
   上記の関数は、コンテキストを作成する際に使用される接続IDを返します。

2. ***OD Create context*関数または*OD Create context dialog*関数を使って、コンテキストを作成する。**
   4th Dimension内のオブジェクトとOracleサーバ上のオブジェクトをバインドするためにコンテキストを使用します。このコンテキストにより、4th Dimensionのフィールドや変数、配列とOracleのカラムやSQL式をバインドすることができます。

   注：4th Dimensionのフィールドの代わりに変数や配列をバインドすることをお勧めします。変数や配列を使用することによりデータ処理を高速に行うことができ、しかもすべてのデータがOracleサーバ上に格納されるので、4th Dimensionのレコードにそのデータのコピーを保存する必要がありません。

   OD Create context関数を使用する場合は、コンテキスト内でバインドを定義するために一連の***OD ADD TO CONTEXT***コマンドを実行する必要があります。サーバ上で選択されるデータを限定または制御するには、***OD SET CLAUSE IN CONTEXT***コマンドを1度ないし数回実行します。例えば、すべてのデータを選択したくない場合は、WHERE句を追加して、ある検索条件を基に選択されたデータに制限することができます。

   OD Create context dialog関数を使用すると、コンテキスト内でバインドやSQL句を定義するためのダイアログボックスが現れます。

   コンテキストの作成に関する詳細は、後述の“ コンテキストの作成 ”の節を参照してください。

3. ***OD Activate context* 関数を使って、コンテキストをアクティブにする。**
   コンテキストをアクティブにすると、4D for OracleはOracleサーバ上のデータを選択するためにSELECT文を生成します。下記のSELECT文は、EMPテーブルから一連のカラムデータを選択します：

   SELECT EMPNO, ENAME,...FROM EMP

   Oracleが問い合わせ(クエリー)を実行すると、4th Dimensionのオブジェクトの中にその結果(リザルト)をロードします。

   コンテキストを1つのテーブルのみで使用している場合は、4th Dimensionの中にロードされたデータを追加、修正、削除することができます。また、ユーザが行った変更内容をOracleサーバに反映します。

   コンテキストのアクティブ処理に関する詳細は、後述の“ コンテキストのアクティブ処理 ”の節を参照してください。

4. **選択されたデータをロードする。*OD Next in context*関数や*OD Previous in context*関数、*OD Goto in context*関数を使って1行づつロードする。または、*OD Load rows context*関数を使って一連の行をロードする。**

データは、バインドによって定義された4th Dimensionのフィールドまたは変数、配列の中にロードされます。データのロードに関する詳細は、後述の“選択データのロード”の節を参照してください。

5. **必要なら、*OD Insert in context*関数、*OD Update in context*関数、*OD Delete in context*関数を使ってデータを追加、修正、削除する。**
   4D for Oracleは、各操作に対するSQL文を生成します。例えば、サーバ上に新規行を追加する場合は、4D for OracleはINSERT文を生成します。

   データの追加、修正、削除に関する詳細は、後述の“コンテキスト内のデータ修正”の節を参照してください。

6. ***OD DEACTIVATE CONTEXT*コマンドを使って、コンテキストを非アクティブにする。**
   非アクティブにされたコンテキストは一時的に使用不可状態になります。しかし、***OD ACTIVATE CONTEXT***コマンドを使用することにより、再度コンテキストをアクティブにすることができます。

7. ***OD DROP CONTEXT*コマンドを使って、コンテキストを閉じる。**
   閉じられたコンテキストは無効になり、使用できなくなります。

8. ***OD LOGOUT*コマンドを使って、接続を閉じる。**

コンテキストの作成

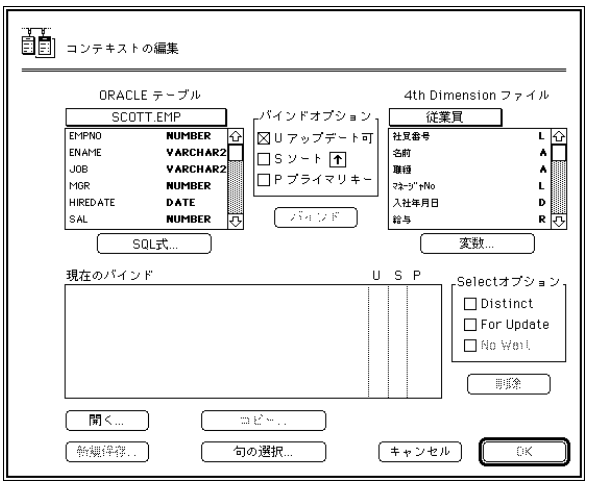
「コンテキストの編集」ダイアログボックスを使ってコンテキストを作成することができます。また、一連のステートメントを実行しても作成することができます。ダイアログボックスを使用すると、4D for Oracleが自動的にコンテキスト定義ステートメントを生成します。

ダイアログボックス使用によるコンテキスト作成

「コンテキストの編集」ダイアログボックスを使って、4th Dimension内のデータ要素にバインドするOracle上のデータを選択するコンテキストを定義することができます。

「コンテキストの編集」ダイアログボックスを開くには、次のように行います：

■ ***OD Create context dialog*関数を実行する。**
   すると、次のような「コンテキストの編集」ダイアログボックスが表示されます。



注：9インチモニタを使用している場合は、このダイアログボックスの表示方法が異なるかもしれません。この場合、4D for Oracleは同じ機能を持った簡素化されたダイアログボックスを表示します。

このダイアログボックスを使用すると、次のようなことを行うことができます：

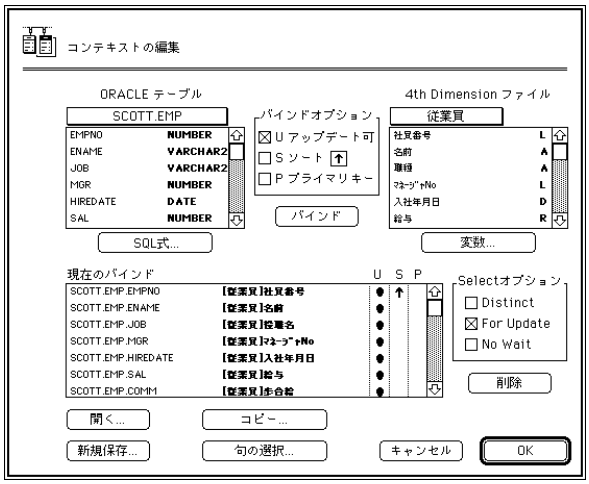
- コンテキスト用のバインドの定義
- コンテキストへのSQL句およびオプションの追加
- クリップボードへのコンテキスト定義コードのコピー
- 任意ファイルへのコンテキスト定義の保存
- 任意ファイルからのコンテキスト定義のロード

バインドの定義

「コンテキストの編集」ダイアログボックスは、ユーザに4th Dimensionのフィールドや変数、配列とOracleのカラムやSQL式を選択させ、その両者をバインドすることができます。

また、複数の4th Dimensionオブジェクトと複数のOracleオブジェクトを結び付けることにより複数のバインドを作成することができます。作成される各バインドは、「現在のバインド」エリアに一覧表示されます。

「コンテキストの編集」ダイアログボックスを使って、4th DimensionオブジェクトとOracle



オブジェクト間のバインドを定義するには、次のように行います：

1. **「Oracleテーブル」ポップアップメニュー下のリストからカラムまたはSQL式を選択する。**
   別のテーブルからカラムを表示するには、「Oracleテーブル」ポップアップメニューからそのテーブルの名前を選択します。

   以前に定義したSQL式の名前を表示するには、「Oracleテーブル」ポップアップメニューから「SQL式」を選択します。「SQL式...」ボタンをクリックして、表示される「SQL式の編集」ダイアログボックスにSQL式を入力することによりSQL式を定義することができます。

2. **「4th Dimensionテーブル」ポップアップメニュー下のリストから4th Dimensionのフィールドまたは変数、配列を選択する。**
   別のテーブルからフィールドを表示するには、「4th Dimensionテーブル」ポップアップメニューからそのテーブルの名前を選択します。

   以前に定義した変数または配列の名前を表示するには、「4th Dimensionテーブル」ポップアップメニューから「変数」を選択します。

   「変数...」ボタンをクリックして、表示されるダイアログボックスに変数名または配列名を入力することにより変数および配列を定義することができます。現在、バインドしたい4th DimensionオブジェクトとOracleオブジェクトが選択されているはずです。コンテキストにバインドを追加する前に、下記の3つの中からバインドオプションを選択することができます。

3. **必要なら、「U アップデート可」や「S ソート」、「P プライマリキー」オプションを選択する。**

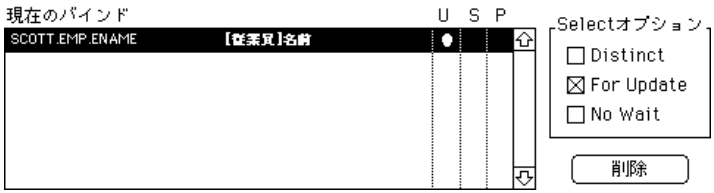
■ **「U アップデート可」：**このオプションでは、***OD Insert in context***関数または***OD Update in context***関数を使って更新されるOracleのカラムを指定します。このオプションはOracleのカラムを含んでいるバインドのみ適用します。SQL式を更新することはできません。

■ **「S ソート」：**このオプションでは、ソートされるSQL式またはカラム内のデータを指定します。このオプションを選択してソート順矢印をクリックすることにより、そのデータを昇順または降順で選択することができます。

■ **「P プライマリキー」：**このオプションではテーブル内の行を重複不可にするカラムを指定します。コンテキスト内でデータを更新するには、データのプライマリキーを指定する必要があります。このオプションは、Oracleのカラムを含んでいるバインドのみ適用します。

4. **「バインド」ボタンをクリックする。**
   これで、コンテキスト内にバインドが設定されます。このバインドはコンテキストがアクティブになるまで実行可能になりません。

5. **コンテキストに追加バインドを定義する場合は、ステップ1~4を繰り返す。**
   すべてのバインドは、バインド用に設定したオプション(「U アップデート可」,「S ソート」,「P プライマリキー」)を表すマーク付きで「現在のバインド」エリアに一覧表示されます。次の図では、EMPテーブルの「ENAME」カラムが[従業員]テーブルの「名前」フィールドにリンクされています。

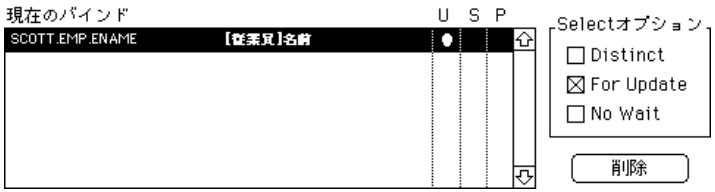


   いつでも、「現在のバインド」エリア内で定義されたバインドを選択し「削除」ボタンをクリックすることにより、バインドを削除することできます。

コンテキストオプションの選択

コンテキストをアクティブにする前に、コンテキストの機能に影響を及ぼすオプションまたはSQL句を選択します。コンテキストオプションを選択するには：

■ **「Selectオプション」エリアからコンテキストオプションを選択する。**



■ **「Distinct」オプション：**このオプションでは、返される値が固有なもののみ指定します。これにより、返される値が複製されないようにします。

■ **「For Update」オプション：**このオプションは、他のすべてのユーザに対して選択された行をロックして、更新される選択行の準備を行います。

■ **「No Wait」オプション：**このオプションは、4D for Oracleがロック解除されたSELECT文によって選択された行を待機しなくてもよいように指定します。もし、その行がロックされていると、問い合わせ(クエリー)は取り消されます。そのため、「No Wait」オプションを選択する前に「For Update」オプションを選択する必要があります。

コンテキストへのSQL句追加

コンテキストにSQL句を追加するには：

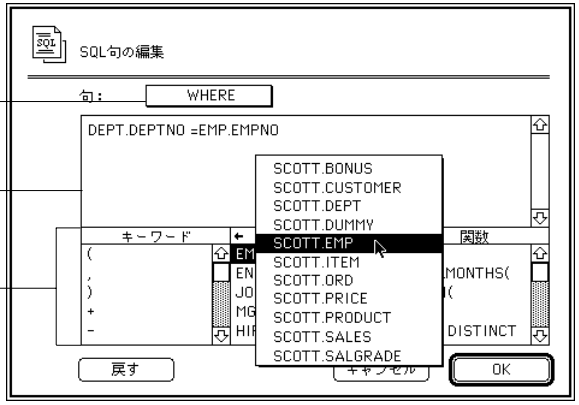
■ **「句の選択...」ボタンをクリックする。**

すると、「SQL句の編集」ダイアログボックスが表示され、“ WHERE ”や“ GROUP BY ”, “ CONNECT BY ”等のSQL句を作成することができます。

ポップアップメニューからSQL句のタイプを選択する

テキストエリアにSQL句を入力する

リストからキーワード、カラム、関数を選択する



「句：」ポップアップメニューからSQL句を選択することにより、作成したいSQL句のタイプを選択することができます。

SQL句を構築するには、テキストエリアの中にSQL句を入力します。任意のSQL句の作成を容易にするには、そのダイアログボックスの下にあるリストからキーワードやカラム名、コマンドを選択します。

中央にあるリストから別テーブルのカラムを表示するには、そのリストのタイトルバー上でマウスボタンを押したままにした際に現われるポップアップメニューからテーブル名を選択するか、または、タイトルバーの両側にある矢印を使ってテーブルリストをスクロールします。

コンテキスト作成ステートメントのコピー

「コンテキストの編集」ダイアログボックスを使ってコンテキストを定義すると、コンテキスト定義と同等の4D for Oracleステートメントを生成する4D for Oracleに問い合わせすることができます。毎回、このダイアログボックスを使ってコンテキストを再定義する代わりに、“ Startup ” メソッドのような4th Dimensionのメソッド内に4D for Oracleステートメントを置くことができます。

コンテキストをアクティブにする前にクリップボードにこれらのステートメントをコピーするには、次のように行います：

1. **「コピー...」ボタンをクリックする。**
   このボタンをクリックすると、4th Dimensionはコンテキスト IDを格納するために使用する4th Dimensionの変数名を尋ねてきます。
2. **変数名を入力し、「OK」ボタンをクリックする。**
   ステートメントがクリップボードにコピーされます。これで、このステートメントを4th Dimensionのメソッド内にペーストすることができます。

ファイルへのコンテキスト定義の保存

4D for Oracleはファイルにコンテキストを保存することができます。ファイルに保存されたコンテキストは、後で一度に「コンテキストの編集」ダイアログボックスの中にロードされます。また、***OD Open context file***関数を使って、メモリ内にロードすることもできます。

ファイルに保存されたコンテキストを保存するには、次のように行います：

1. **「新規保存...」ボタンをクリックする。**
   「ファイル保存」ダイアログボックスが表示され、保存するファイル名とそのファイルを格納する場所を指定することができます。
2. **「保存」ボタンをクリックする。**
   ファイルが保存されます。

ファイルからのコンテキストのロード

前節でファイルに保存したコンテキストをロードすることができます。

ファイルに保存したコンテキストをロードするには、次のように行います：

1. **「コンテキストの編集」ダイアログボックス内の「開く...」ボタンをクリックする。**
   「ファイルオープン」ダイアログボックスが表示されます。
2. **ファイルを選択し「開く...」ボタンをクリックする。**
   コンテキストはロードされ、「コンテキストの編集」ダイアログボックス内に表示されます。

   また、***OD Open context file***関数を使ったメソッドでコンテキストをロードすることもできます。

ステートメントを使ったコンテキストの作成

「コンテキストの編集」ダイアログボックスを使用したくない場合は、4D for Oracleのコマンドや関数を使ったメソッドでコンテキストを定義することができます。

メソッドでコンテキストを作成する際の基本ステップは、次のようになります：

1. ***OD Create context*関数を実行して、コンテキストを初期化しコンテキストIDを取り出す。**
   コンテキストIDは、すべてのコンテキストコマンドでコンテキストを確認するために使用されます。

2. **一連の*OD ADD TO CONTEXT*ステートメントを実行して、コンテキストを定義する。**
   OD ADD TO CONTEXTコマンドは、4th Dimensionのオブジェクトと Oracleのオブジェクトをバインドしたり、「U アップデート可」,「S ソート」.「P プライマリキー」オプションと同等の機能を追加することができます。ここで、4D for Oracleによって構築されるSELECT文は次のようになります：

   SELECT EMPNO, ENAME,... FROM EMP

3. **一連の*OD SET CLAUSE IN CONTEXT*ステートメントを実行して、コンテキスト定義にSQL句を追加する。**
   もし、部門コード20の従業員だけを参照したい場合は、次のようなステートメントを使用します。

   OD SET CLAUSE IN CONTEXT **(Context_ID ; 2 ; "DEPTNO=20")**

   この行を追加して、問い合わせ(クエリー)を次のように修正します：

   SELECT EMPNO, ENAME,... FROM EMP WHERE DEPTNO=20

   “ WHERE ” キーワードは、自動的に4D for Oracleによって追加されます。

   一連の***OD Get clause in context***ステートメントを実行して、あるコンテキストのSQL句のセットを取り出すことができます。

コンテキストのアクティブ処理

コンテキスト定義によって生成されるSELECT文を送信するには、コンテキストをアクティブにする必要があります。コンテキストをアクティブにするには、次のように行います：

- ***OD Activate context***ステートメントを実行する。
  そのコンテキストに対応しているデータが、サーバ上で選択されます。

  4th Dimension内で選択されたデータを使用するには、フィールドや変数、配列の中にそのデータをロードする必要があります。

選択されたデータのロード

Oracleサーバ上でデータが選択されると、4th Dimensionのフィールドや変数、配列の中にそのデータをロードすることによりそれを表示することができます。データをロードする場合、1回に1つの行または同時に複数の行をロードするために選択することができます。

1行のロード

1つの行をロードするには、***OD Next in context***関数や***OD Previous in context***関数、***OD Goto in context***関数を実行します。例えば、あるコンテキスト内で次の行を表示するには、次のようなステートメントを実行します：

$res:=*OD Next in context* (myContext)

行は、コンテキストの中で指定された4th Dimensionのオブジェクト内に表示されます。ロードしたい行の番号を知りたい場合は、次のようなステートメントを使用します：

$res:=*OD Goto in context* (myContext ; 12)

これで指定された行をロードすることができます。

複数行のロード

同時に複数の行をロードするには、***OD Load rows in context***ステートメントを実行します。この場合、4D for Oracleはすべての行をロードし、必要なレコードまたは配列を作成します。

必要なら、同時にロードできる行の最大数を制限することができます。例えば、次のステートメントはコンテキストによって選択される最初の100行をロードします：

$res:=*OD Load rows in context* (myContext ; 100)

クライアントマシン上でデータのコピーを保存しないようにするために、変数または配列の中にデータをロードすることをお勧めします。多くのアプリケーションでは、ほとんどのデータはOracleサーバ上で保存されます。

コンテキスト内でのデータ修正

コンテキスト内で行(rows)を追加、修正、削除することができます。コンテキストコマンドを使ってサーバ上のデータを操作する場合、4D for Oracleは4th Dimension内のカレントデータを基に追加、修正、削除されるデータを知っています。

行の追加

OD Insert in context関数を使って、新規行を追加する場合、4D for Oracleはサーバ上に新規行を作成するために現在の4th Dimensionオブジェクトの中にあるデータを使用します。例えば、あるレコードがサーバ上の新規行としてフィールドに入力された情報を追加するために登録される際、***OD Insert in context***ステートメントが実行されます。

行の更新

OD Update in context関数を使って、ある行を更新する場合、4D for Oracleはサーバ上の対応している行を修正するためにコンテキストによって指定された4th Dimensionオブジェクトの中にあるデータを使用します。例えば、サーバ上の対応しているカラム(columns)を修正するためにあるレコード内で何かが修正される時は必ず、***OD Update in context***ステートメントが実行されます。

行の削除

OD Delete in context関数を使って、ある行を削除すると、4D for Oracleは4th Dimensionのカレントオブジェクトに対応しているサーバ上の行を削除します。例えば、サーバ上の対応している行を削除するためにあるレコードが削除される際、***OD Delete in context***ステートメントが実行されます。

コンテキストの非アクティブ処理とクローズ処理

コンテキストへの変更を行うために、一時的にコンテキストを非アクティブにします。例えば、コンテキスト内のSQL句を追加または修正するために任意のコンテキストを非アクティブにしたい場合があるかもしれません。コンテキストを非アクティブにするには、***OD DEACTIVATE CONTEXT***コマンドを使用します。

コンテキストの使用を終了すると、コンテキストを閉じることができます。コンテキストを閉じることにより、以前、メモリ内でコンテキスト定義によって使用された領域を解放することができます。閉じられたコンテキストを再度、アクティブにすることはできません。コンテキストを閉じるには、***OD DROP CONTEXT***コマンドを使用します。

OD DROP CONTEXTコマンドは、コンテキストがまだ非アクティブになっていない場合にそのコンテキストを非アクティブにします。4th Dimensionが終了すると、4D for Oracleは自動的にすべてのコンテキストを閉じます。

コンテキストとカーソル

コンテキストコマンドを使ってデータを操作する場合、4D for Oracleはサーバ上でのカーソル作成を透過的に管理します。「カーソル」は、いつでもOracleによって作成される単一SQL文のステータスの要約情報を含んだメモリバッファのことです。

コンテキストのアクティブ処理は、自動的にカーソルの作成をトリガーします。このカーソルを「選択カーソル」と呼びます。もし、サーバ上のデータを追加、修正、削除すると、別のカーソルが作成されます。このカーソルを「更新カーソル」と呼びます。

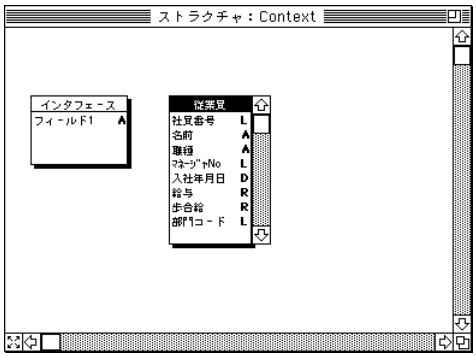
コンテキストチュートリアル

このチュートリアルでは、4D for Oracleで提供されている「context」データベースを使って、データを管理するためのコンテキスト使用の基本を学習します。このデータベースのストラクチャは、Oracleの例題として提供されているEMPテーブルと同じです。

チュートリアルデータベースのインストールとオープン

下記のインストールステップに従い、「Context」データベースを開く。

1. **ハードディスクに「Context」データベースをコピーする。**
2. **「Context」データベースに4D for Oracleをインストールする。**
3. **4th Dimensionまたは4D Clientを使って、このデータベースを開く。**
   すると、次のようなストラクチャが表示されます。



このデータベースには[従業員]テーブルと[インタフェース]テーブルの2つが含まれており、あらかじめ入出力のフォームが作成されています。また、4D for Oracleのステートメントを含んだ4th Dimensionのメソッドが組み込まれています。

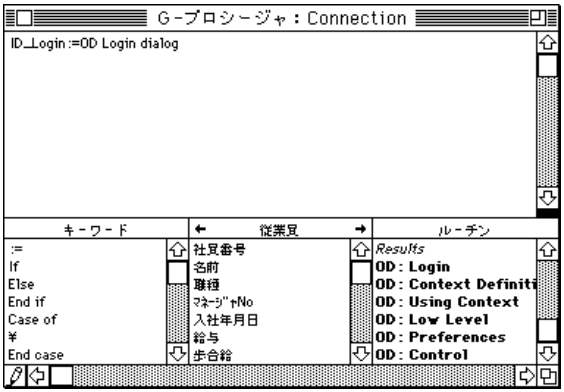
ここでは、4th Dimensionを使用して“ scott ”というOracleのログイン名で例題テーブルのデータを調べます。

接続の設定

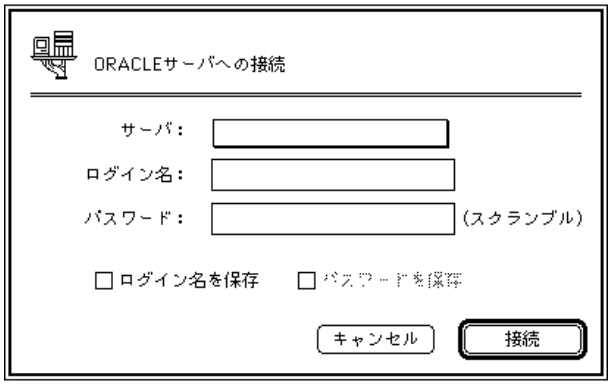
ここで、4th DimensionとOracleの間の接続を設定する必要があります。しかし、このデータベースには、すでに接続メソッドが作成されてます。

1. **「デザイン」メニューから「メソッド」を選択する。**
2. **「Connection」メソッドを開く。**
   メソッドが4th Dimensionの「メソッド」エディタに表示されます。4D for Oracleによってランゲージに追加されるコマンドや関数は、「メソッド」エディタ内の「ルーチン」リストの最後でカテゴリー別にグループ化されます。
   「ルーチン」リストの最後をスクロールすると、4D for Oracleルーチンが表示され、確認することができます。すべての4D for Oracleルーチンは、“ OD ”という文字が接頭辞に付きます。

OD Login dialog関数は、標準ダイアログボックスを表示します。このダイアログボックスにより、接続したいサーバを選択し、ユーザのログイン名とパスワードを入力することができます。この関数は「ルーチン」リストの「Login」カテゴリー内にあります。



3.「ユーザ」モードに移動し、「Connection」メソッドを実行する。
「Oracle サーバへの接続」ダイアログボックスが表示されます。

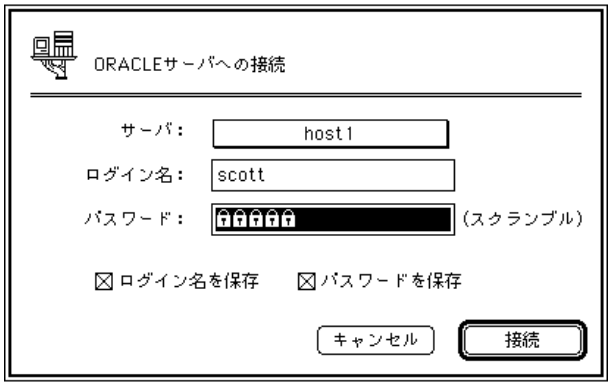


4.「サーバ」ポップアップメニューから接続したいサーバを選択する。
このポップアップメニューは、「TNSNAMES.ORA」ファイルに記述されているサーバを含んでいます。このファイルは、Windows下では“ Orawin/Network/admin ”ディレクトリ、MacOS下では“ Oracle home:network:admin ”フォルダの中にあります。「TNSNAMES.ORA」ファイルの書式に関する詳細は、Oracleのマニュアルを参照してください。もし、MacOS下で OCI6を使って4D for Oracleを使用している場合は、その Oracle homeフォルダ内の「config.ora」ファイルに記述されているサーバを含んでいます。

5. ログイン名として“ scott ”を入力する。

6.「パスワード」ボックスに scottのパスワードを入力する。

scottのパスワードは、Oracleサーバのインストール作業に何も変更を加えていない場合は、“ tiger ” です。



4D for Oracleは、「ログイン名を保存」チェックボックスが選択されている場合、ユーザのログイン名を保存します。パスワードに関しても同様の処理を行ないます。

ログイン名とパスワードは「システムフォルダ」内にある「4D for Oracle Preferences」初期設定ファイルに保存され、次回の接続時に使用されます。

注：別のユーザがあなたのマシンへのアクセス権を持っていると、そのユーザは「ログイン名を保存」および「パスワードを保存」チェックボックスが選択されていれば、あなたのログイン名とパスワードを知ることができます。もし、機密保持を厳密に行いたい場合は、これらのチェックボックスを選択しないようにしてください。

7. **接続の準備が完了したら、「接続」ボタンをクリックしてOracleサーバに接続する。**接続したくない場合は、「キャンセル」ボタンをクリックします。
エラーが何も発生しなければ、***OD Login dialog***関数は変数 “ ID_Login ” に接続IDを返します。後で、接続を作成する際にこの接続IDを使用します。

   接続エラーが発生すると、この関数は-1を返します。「キャンセル」ボタンをクリックすると、0を返します。

   同じサーバでしかも同じユーザアカウント(ログイン名)で複数の接続を作成することができます。異なる接続IDが各接続ごとに返されます。この場合、ユーザはおそらく1回のログインだけで接続できるようにしたくなるでしょう。これを行うには、複数の接続を避けるために下記のように「Connection」メソッドを書き換えます。

```
If (OD Login state (ID_Login) >0)
    ALERT ("あなたは、すでに接続されています！")
Else
    ID_Login:=OD Login dialog
End if
```

   OD Login state関数は、接続ステータスを返します。そして、接続ID、つまり “ ID_Login ” 変数が正数(接続済)または正数以外(非接続)かのテストを行います。接続が切れると接続IDは更新されないので、この***OD Login state***関数の使用は一層効果的になります。

   また、あなたが接続を解除を試みる前に、あなたがサーバに接続されているかどうかをチェックするためのメソッドを書くこともできます。

8. **「デザイン」モードに移動し、「Disconnection」メソッドを開く。**

次のメソッドは、サーバへの接続をクローズするために使用されます。

```
If (OD Login state (ID_Login)=0)
    ALERT ("あなたは、接続されていません。")
Else
    OD LOGOUT
End if
```

4D for Oracleは4th Dimensionが終了すると、すべてのサーバへの接続を自動的にクローズします。

コンテキストの作成

「Connection」メソッドを実行すると、Oracleサーバに接続できます。

Oracleデータを使って処理を行うために、SCOTTテーブルの1つと4th Dimensionのテーブルをバインドします。これを行う最も簡単な方法は、コンテキストを使用することです。コンテキストを使ってデータ処理を行うと、Oracleのカラム(columns)とフィールドや変数、配列等の4th Dimensionのオブジェクトをリンクすることができます。

ここでは、ダイアログボックスを使ってコンテキストを定義することができるメソッドを実行します。コンテキストが定義されると、そのメソッドは自動的に行(rows)をロードして、4th Dimensionのファイルにそれを表示します。

1. 「デザイン」モードにおいて、「Context Demo」メソッドを開く。

このメソッドは、Oracleのテーブルから一連のデータをロードし、4th Dimensionのテーブルにそれを保存します。

```
`ARRAY STRING (30;tLocation;0)
`ARRAY STRING (30;tJob;0)
`ARRAY REAL (tSalaries;0)
If (OD Login state (ID_Login)>0)                `接続が有効かどうかチェックする
   `「コンテキストの編集」ダイアログボックスを表示する
   ID_Context:=OD Create context dialog (ID_Login;"SCOTT.EMP";File(>>[従業員]))
   If (ID_Context >0)                           `コンテキストが定義されている場合
      $rc:=OD Activate context (ID_Login;ID_Context)`コンテキストをアクティブにする
      ALL RECORDS ([従業員])
      DELETE SELECTION ([従業員])
      $n:=OD Load rows context (ID_Context)  `コンテキストによって選択された行をロード
      ALL RECORDS ([従業員])
      ORDER BY ([従業員];[従業員]社員番号 ; >) `レコードをソートする
      OD DEACTIVATE CONTEXT (ID_Context)
      OD DROP CONTEXT (ID_Context)
   End if
End if
```

OD Create context dialog関数は、4th DimensionオブジェクトとOracleオブジェクト間のバインドを作成することができるダイアログボックスを表示します。***OD Create context dialog***関数は、ダイアログボックス内の「キャンセル」ボタンがクリックされない場合や処理がうまくいった場合に正数の接続IDを返します。この接続IDは、その後のコマンドの中で使用されます。

作成されたコンテキストは、4th DimensionオブジェクトとOracleオブジェクト間のバインドを定義します。***OD Activate context***関数をコールして、コンテキストをアクティブにします。***OD Activate context***関数は選択カーソルを作成し、4D for Oracleに「コンテキストの編集」ダイアログボックスに入力された情報からバインドするSELECT文を実行させます。

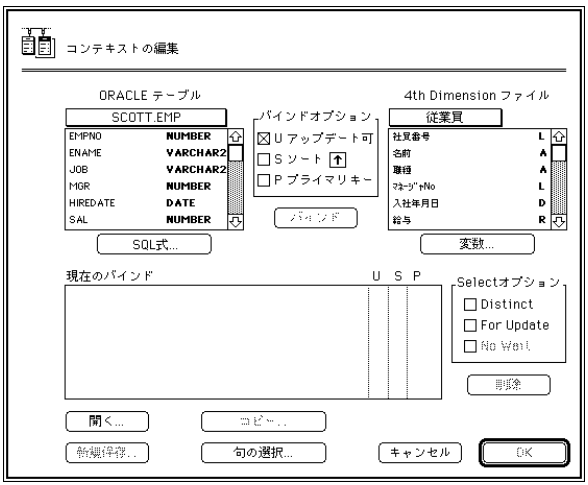
注：「コンテキストの編集」ダイアログボックスが取得した接続IDは、Oracleサーバからテーブルとカラムの名前を検索する場合にしか有効ではありません。しかし、もし、コンテキスト定義中に指定されたOracleのテーブルがアクティブ処理用に使用される接続からアクセス可能になっていれば、コンテキストがアクティブになっている間、別の接続IDを使用することができます。

OD Load rows context関数は、4th Dimensionの[従業員]テーブルにOracleの“ SCOTT.EMP ”テーブルの内容を送信します。データを転送する前に、テーブル内を新しく転送されるレコードだけにするために***DELETE SELECTION***コマンドが使用されています。

OD DEACTIVATE CONTEXTコマンドは、***OD Activate context***関数によって開かれたカーソルを閉じます。コンテキストによって定義されたリンク情報は、非アクティブになります。しかし、***OD Activate context***関数を使って、再度それらをアクティブにすることができます。

OD DROP CONTEXTコマンドは、コンテキストで使用したメモリを解放します。このコマンドをコールした後は、コンテキストを再定義または再度アクティブにするまで、そのコンテキストを再利用することはできません。

2.「ユーザ」モードに移動し、「Context Demo」メソッドを実行する。
「コンテキストの編集」ダイアログボックスが表示されます。



ダイアログボックス上部にある2つのポップアップメニューで使用したいOracleのテーブルと4th Dimensionのテーブルを選択することができます。

OD Create context dialog関数を呼び出すと、Oracleのデフォルトテーブルとして“ SCOTT.EMP ”、4th Dimensionのデフォルトテーブルとして[従業員]が指定されています。そして、“ SCOTT.EMP ”テーブルのカラムと[従業員]テーブルのフィールドがあらかじめ一覧表示されています。

この例題では、各データに「U アップデート可」バインドオプションを使用します。

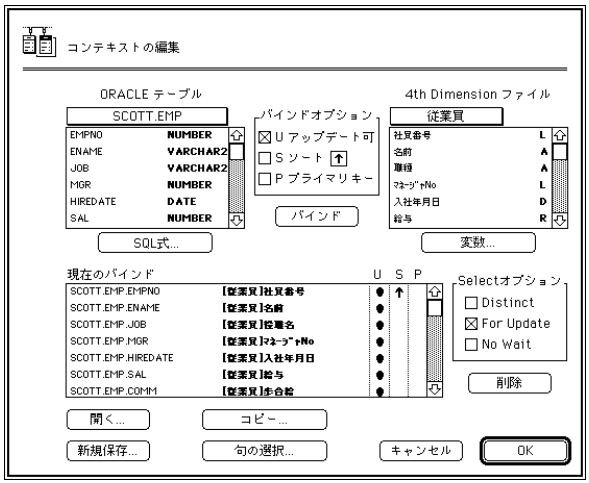
3. **「EMPNO」カラムと「社員番号」フィールドを選択する。**

4. **「U アップデート可」と「S ソート」チェックボックスを選択する。**
   「U アップデート可」チェックボックスを選択することにより、「EMPNO」カラム内の値を更新したいことを示します。「S ソート」オプションが設定されていれば、4D for Oracleはこのカラム内の値をロックします。データの更新方法に関しては、後述の“ サーバ上でのデータの追加と修正 ”の節で学習します。

   「S ソート」チェックボックスを選択することにより、「EMPNO」カラム内のデータがソート順に並べられたことを指定します。デフォルトのソート順位は昇順です。もし、必要なら、「S ソート」チェックボックスの右側にある矢印マークをクリックして、降順にソート順位を変更することができます。

5. **「EMPNO」カラムと「社員番号」フィールドをバインドするために「バインド」ボタンをクリックする。**
   すると、バインド情報が「現在のバインド」エリアに一覧表示されます。

6. **残りのカラムとそれに対応する[従業員]テーブル内のフィールドをバインドし、それぞれのバインド用に「U アップデート可」チェックボックスを選択する。**
   現在、「コンテキストの編集」ダイアログボックスは次のようになっているはずです。

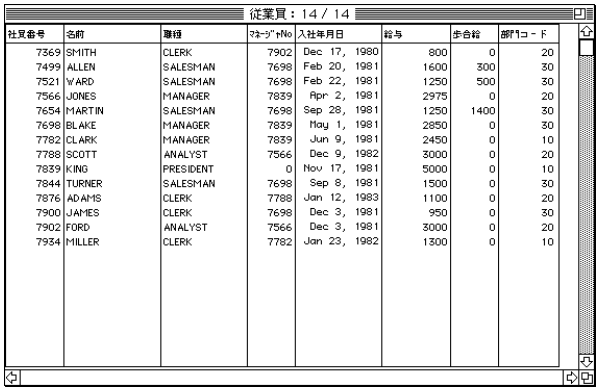


7. **「新規保存...」ボタンをクリックして、ファイルにこのコンテキストを保存する。**
   次のように保存するファイル名を尋ねるダイアログボックスが表示されます。

「保存」ボタンをクリックすると、ディスク上にコンテキストが保存されます。この保存されたコンテキストは、このチュートリアルの後半部分で使用されます。

8.「コンテキストの編集」ダイアログボックス内の「OK」ボタンをクリックして、定義したコンテキストを有効にする。

4D for Oracleはこのコンテキストをアクティブにして、[従業員]テーブルの中に“ SCOTT.EMP ”テーブルの行をロードします。



従業員：14 / 14

社員番号	名前	職種	マネージャNo	入社年月日	給与	歩合給	部門コード
7369	SMITH	CLERK	7902	Dec 17, 1980	800	0	20
7499	ALLEN	SALESMAN	7698	Feb 20, 1981	1600	300	30
7521	WARD	SALESMAN	7698	Feb 22, 1981	1250	500	30
7566	JONES	MANAGER	7839	Apr 2, 1981	2975	0	20
7654	MARTIN	SALESMAN	7698	Sep 28, 1981	1250	1400	30
7698	BLAKE	MANAGER	7839	May 1, 1981	2850	0	30
7782	CLARK	MANAGER	7839	Jun 9, 1981	2450	0	10
7788	SCOTT	ANALYST	7566	Dec 9, 1982	3000	0	20
7839	KING	PRESIDENT	0	Nov 17, 1981	5000	0	10
7844	TURNER	SALESMAN	7698	Sep 8, 1981	1500	0	30
7876	ADAMS	CLERK	7788	Jan 12, 1983	1100	0	20
7900	JAMES	CLERK	7698	Dec 3, 1981	950	0	30
7902	FORD	ANALYST	7566	Dec 3, 1981	3000	0	20
7934	MILLER	CLERK	7782	Jan 23, 1982	1300	0	10

ここで、Oracleのデータを使って検索やソートを実行したり、またはクイックレポートやラベルを作成するために4th Dimensionを使用することができます。

複合問い合わせ(クエリー)の作成

前節では、コンテキストを定義し、それをアクティブにしました。コンテキストがアクティブにされると、4D for Oracleは次のような簡単なSQL問い合わせ(クエリー)を生成します：

```
SELECT ENAME,JOB,MGR, ... FROM EMP
```

すると、カラム結果が対応している4th Dimensionのフィールド内にロードされます。

有効なSQL問い合わせ(クエリー)のカラム結果と4th Dimensionのフィールドや変数、配列を関係付けることができます。例えば、給与別にソートされた各部門における各職種別の平均給与を問い合わせることができます。これのSQL問い合わせ(クエリー)は、次のようになります。：

```
SELECT EMP.JOB,DEPT.LOC,AVG( EMP.SAL )
FROM EMP,DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY DEPT.LOC,EMP.JOB
ORDER BY 3
```

複合クエリーの作成

ここでは、このSQL問い合わせ(クエリー)を実行するコンテキストを定義します。そして、配列「tLocation」,「tJob」,「tSalaries」の中に3つのカラム結果をロードします。

1. 再度、「Context Demo」メソッドを開き、そのメソッド内の先頭から3行目までのコメント記号(`)を削除する。

```
ARRAY STRING (30 ; tLocation ; 0)
ARRAY STRING (30 ; tJob ; 0)
ARRAY REAL (tSalaries ; 0)
```

前記の3行で、Oracleのデータを取り込むための3つの配列を宣言します。「tLocation」と「tJob」は30バイトの文字列配列で、一方の「tSalaries」は実数配列です。4D for Oracleは、自動的にSQL問い合わせ(クエリー)が返す行の数にこれらの配列をサイズ変更します。

2.「ユーザ」モードに移動し、再度「Context Demo」メソッドを実行する。
「コンテキストの編集」ダイアログボックスが表示されます。

「JOB」カラムと配列「tJob」のバインド

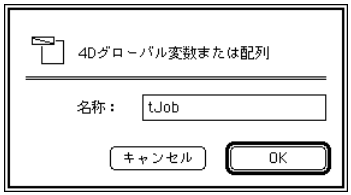
このバインドを作成するには：

1. 左側のリストから「JOB」カラムを選択する。

2.「変数...」ボタンをクリックする。
すると、「グローバル変数または4D配列」ダイアログボックスが表示されます。このダイアログボックスにおいて、「JOB」カラム内のデータを取り込むための4th Dimension配列の名前を入力します。

3.“ tJob ”と入力し、「OK」ボタンをクリックする。

4.「バインド」ボタンをクリックする。
4D for Oracleは、「JOB」カラムと「tJob」配列のバインドを定義します。

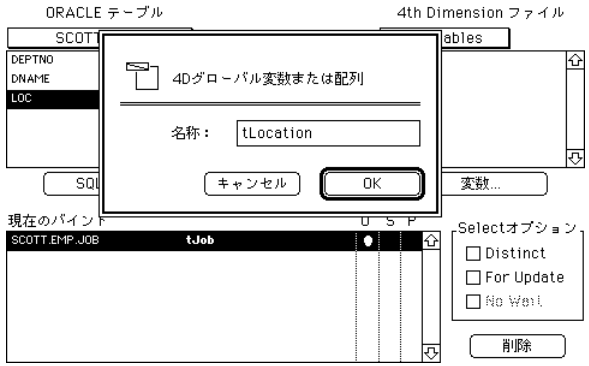


「LOC」カラムと「tLocation」配列のバインド

このバインドを作成するには：

1.「Oracleテーブル」ポップアップメニューから「SCOTT.DEPT」テーブルを選択し、「LOC」カラムを選択する。

2.「変数...」ボタンをクリックし、“ tLocation ”と入力して「OK」ボタンをクリックする。



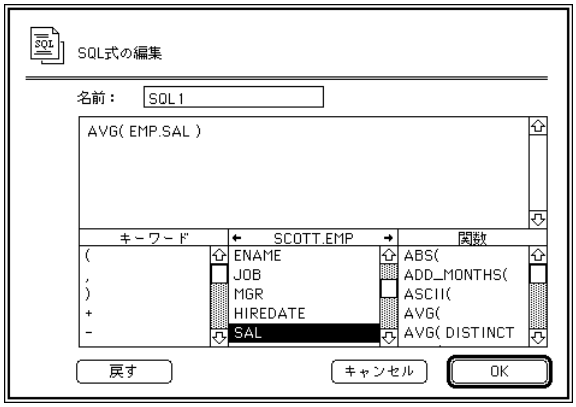
3.「バインド」ボタンをクリックする。
4D for Oracleは、「LOC」カラムと「tLocation」配列のバインドを定義します。

SQL式「SQL 1」と「tSalaries」配列のバインド

このバインドを作成するには：

1.「SQL式...」ボタンをクリックし、「SQL式の編集」ダイアログボックス内の編集エリアに“ AVG(EMP.SAL) ”と入力して、「OK」ボタンをクリックする。

このダイアログボックス内で有効なSQL式を入力することができます。



デフォルトでは、SQL式は「SQL 1」と名付けられています。

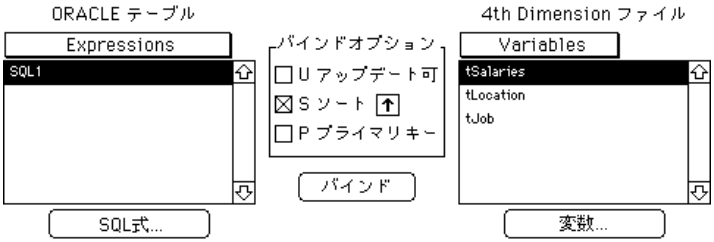
2.「Oracleテーブル」ポップアップメニューから「SQL 1」を選択する。

SQL式の名前が、そのポップアップメニュー下部のリストエリアに表示されます。

3. 左側のリストから「SQL 1」を選択する。

4.「変数...」ボタンをクリックし、“ tSalaries ”と入力して「OK」ボタンをクリックする。

5.「S ソート」チェックボックスを選択して、ソートするカラムを指定する。



6.「バインド」ボタンをクリックする。

4D for Oracleは、SQL式「SQL 1」と「tSalaries」配列の間のバインドを定義します。

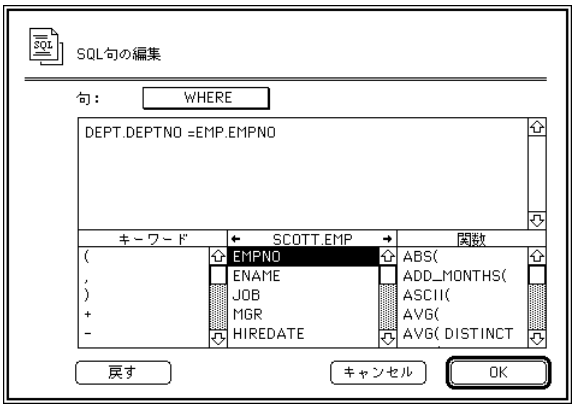
ここで、このコンテキストの対する「WHERE」句と「GROUP BY」句を指定します。

このコンテキストの対する「WHERE」句と「GROUP BY」句をの定義

SQL句を定義するには：

1.「句の選択...」ボタンをクリックする。

「SQL句の編集」ダイアログボックスが表示されます。

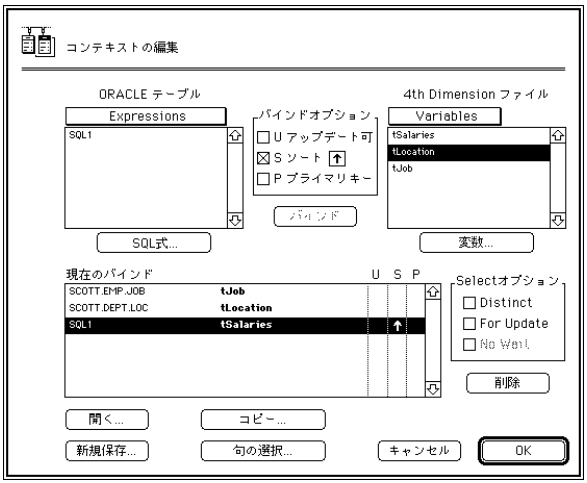


2. **「SQL句」ポップアップメニューから「WHERE」を選択し、“ DEPT.DEPTNO=EMP.DEPTNO ” と入力する。**

3. **「句」ポップアップメニューから「GROUP BY」を選択し、“ DEPT.LOC,EMP.JOB ” と入力する。**

4. **「OK」ボタンをクリックして、定義したSQL句を受け付ける。**

   すると、次のように「コンテキストの編集」ダイアログボックスが表示されます。



5. **「OK」ボタンをクリックしコンテキストを有効にして、「Context Demo」メソッドの実行を再開する。**

   4D for Oracleは定義した問い合わせ(クエリー)を実行し、配列「tJob」,「tLocation」,「tSalaries」の中にその実行結果をロードします。

6. **「Results」メソッドを実行して、結果を確認する。**

   すると、3つの配列を表示する「問い合わせ結果」ダイアログボックスが表示されます。

メソッドを使ったコンテキストの作成

高度なアプリケーションでは、「コンテキストの編集」ダイアログボックスはSQLを知らないユーザに見せないようにする必要があります。4D for Oracleには、メソッドでコンテキストを作成する方法をいくつか用意しています。

コンテキストを作成するメソッドの生成

コンテキストを作成するメソッドを生成するには、次のように行います：

1. **「ユーザ」モードに移動し、再度「Context Demo」メソッドを実行する。**
   「コンテキストの編集」ダイアログボックスが表示されます。

2. **「開く...」ボタンをクリックし、先ほど保存したコンテキストファイルをロードする。**
   4D for Oracleは、このダイアログボックス内でコンテキスト定義を再作成します。

3. **「コピー...」ボタンをクリックし、変数名を " ID_Context " として「OK」ボタンをクリックする。**
   4D for Oracleはコンテキスト定義に必要なメソッドを生成し、クリップボードにその内容をコピーします。変数「ID_Context」は接続IDとして使用されます。

4. **「デザイン」モードに移動し、「Context Demo」メソッドを開く。**

5. **前節の例題で使用した配列宣言を行う3つのコードを削除する。**

6. ***OD Create context dialog*ステートメントの箇所を選択し、「編集」メニューから「ペースト」を選択する。**
   すると、OD Create context dialogステートメントの箇所が「コンテキストの編集」ダイアログボックスによって作成されたコードに置き換わります。現在、このメソッドは次のようになっているはずです。

```
If (OD Login state (ID_Login)>0)
   ID_Context:=OD Create context("SCOTT.EMP")
   If(ID_Context > 0)
      OD ADD TO CONTEXT(ID_Context;"EMP.EMPNO";>>[従業員]社員番号;3)
      OD ADD TO CONTEXT(ID_Context;"EMP.ENAME";>>[従業員]名前;1)
      OD ADD TO CONTEXT(ID_Context;"EMP.JOB";>>[従業員]職種;1)
      OD ADD TO CONTEXT(ID_Context;"EMP.MGR";>>[従業員]ﾏﾈｰｼﾞｬNo;1)
      OD ADD TO CONTEXT(ID_Context;"EMP.HIREDATE";>>[従業員]入社年月日;1)
      OD ADD TO CONTEXT(ID_Context;"EMP.SAL";>>[従業員]給与;1)
      OD ADD TO CONTEXT(ID_Context;"EMP.COMM";>>[従業員]歩合給;1)
      OD ADD TO CONTEXT(ID_Context;"EMP.DEPTNO";>>[従業員]部門コード;1)
   End if
   If (ID_Context>0)     `コンテキストをチェックする
      $rc:=OD Activate context (ID_Login;ID_Context)
      ALL RECORDS ([従業員])
      DELETE SELECTION ([従業員])     `既存レコードを削除する
      $n:=OD Load rows context (ID_Context)     `Oracleデータを取得する
      ALL RECORDS ([従業員])
      ORDER BY ([従業員];[従業員]社員番号;>)  `Oracleデータと同じように
                                                ソートする
      OD DEACTIVATE CONTEXT (ID_Context)
      OD DROP CONTEXT (ID_Context) `このコンテキストはもはや必要ない
   End if
End if
```

これで、このメソッドは「コンテキストの編集」ダイアログボックスを表示することなく、コンテキストをアクティブにします。このメソッドが明確な2つの部分に分かれている点に注目してください。

- 最初の部分では、コンテキストを定義しています。
- 2番目の部分では、1度アクティブになったコンテキストを使用しています。
  このメソッドでは、コンテキストはデータを表示するためにのみ使用されています。もし、データを修正するためにこのコンテキストを使用したい場合は、プライマリ(主)キーとなるカラムを1つ宣言する必要があります。この場合、***OD ADD TO CONTEXT*** コマンドの引数にこの属性を指定することができます。

Startupメソッドでのコンテキストの定義

コンテキストの定義は既存コンテキストを必要としないので、その定義を4th Dimensionの「Startup」メソッド内で行うことができます。

1.「Context Demo」メソッドのコンテキスト定義部分をカットし、それを「Startup」メソッドの中にペーストする。

コンテキストの定義部分は次のステートメントの箇所です。

```
ID_Context:=OD Create context ("SCOTT.EMP")
If (ID_context >0)
   OD ADD TO CONTEXT(ID_Context ; "EMP.EMPNO" ; >>[従業員]社員番号 ; 3)
   OD ADD TO CONTEXT(ID_Context ; "EMP.ENAME" ; >>[従業員]名前 ; 1)
   OD ADD TO CONTEXT(ID_Context ; "EMP.JOB" ; >>[従業員]職種 ; 1)
   OD ADD TO CONTEXT(ID_Context ; "EMP.MGR" ; >>[従業員]ﾏﾈｰｼﾞｬNo ; 1)
   OD ADD TO CONTEXT(ID_Context ; "EMP.HIREDATE" ; >>[従業員]入社年月日 ; 1)
   OD ADD TO CONTEXT(ID_Context ; "EMP.SAL" ; >>[従業員]給与 ; 1)
   OD ADD TO CONTEXT(ID_Context ; "EMP.COMM" ; >>[従業員]歩合給 ; 1)
   OD ADD TO CONTEXT(ID_Context ; "EMP.DEPTNO" ; >>[従業員]部門コード ; 1)
End if
```

何も記述されていない空の「Startup」メソッドが、すでに作成されています。

2.「Context Demo」メソッドから次のコードを削除する。

```
OD DROP CONTEXT (ID_Context)        `このコンテキストはもはや必要ない
```

現在、「Context Demo」メソッドは次のようになっているはずです：

```
If (ID_Context >0)        `コンテキストをチェックする
   $rc:=OD Activate context (ID_Login ; ID_Context)
   ALL RECORDS ([従業員])
   DELETE SELECTION ([従業員])        `既存レコードを削除する
   $n:=OD Load rows context (ID_Context)        `Oracleデータを取得する
   ALL RECORDS ([従業員])
   ORDER BY ([従業員];[従業員]社員番号 ; >)`Oracleデータと同じようにソートする
   OD DEACTIVATE CONTEXT (ID_Context)
   End if
End if
```

3. **「ユーザ」モードに戻り、「Startup」メソッドを実行しコンテキストを定義する。**
   「Startup」メソッドは、データベースが開かれる際に4th Dimensionによって自動的に実行されます。

4. **「Context Demo」メソッドを実行し、そのアプリケーションが同じように操作されるかどうか確認する。**
   出力フォームは、コンテキストによって選択されたレコードの一覧を表示します。

ディスク上に保存されたコンテキストの使用

OD Open context file関数は、ディスクに保存された定義情報を使ってコンテキストを作成します。前節の例題のコンテキスト定義を次のように変更することができます：

```
ID_Context:=OD Open context file ("myContext")
```

「Context Demo」メソッドは、そのままで変わりません。

ピクチャ内に保存されたコンテキストの使用

4th Dimensionのピクチャフィールドまたは変数内にコンテキストの定義情報を格納することができます。これを行うには、ピクチャタイプのオブジェクトを返す***OD Save context picture***関数を使用します。前節の例題のコンテキスト定義を次のように変更することができます：

```
ID_Context:=OD Open context picture ("myContext")
```

「Context Demo」メソッドは、そのままで変わりません。

サーバ上でのデータの追加と修正

これまでは、4th Dimensionの中へのデータのロード方法について学習してきました。しかし、Oracleサーバ上でのデータの更新方法についてはまだ学習していません。そこで、ここでは、Oracleサーバ上にある“ SCOTT.EMP ”テーブルでのデータの修正方法について学習することにします。例題データベースの中にある3つのメソッドを使用します。

コンテキストのアクティブ処理

「Activate」メソッドはコンテキストをアクティブにして、選択された行を出力フォームに表示します。

```
If (OD Login state (ID_Login) >0)
   If (OD Context state (ID_Context) >0)
      $rc:=OD Activate context (ID_Login ; ID_Context)
      ALL RECORDS ([従業員])
      DELETE SELECTION ([従業員])
      $rc:=OD Load rows context (ID_Context)
      ALL RECORDS ([従業員])
      ORDER BY ([従業員];[従業員]社員番号 ; >)
   End if
End if
```

コンテキストの非アクティブ処理

「Deactivate」メソッドはコンテキストの有効チェックを行って、そのコンテキストを非アクティブにします。

```
If (OD Context state (ID_Context)>1)
   OD DEACTIVATE CONTEXT (ID_Context)
End if
```

レコードの作成と修正

「ManageContext」メソッドは、レコードを作成したり修正することができます。

```
If (OD Context state (ID_Context)>1)        `コンテキストが有効かどうかチェックする
   Case of
   \ (Before)
    fCreation:=Modified record ([従業員])
    If( Not(fCreation))
      If (Selected record number ([従業員])<=OD Records in context (ID_Context))
      $rc:=OD Goto in context (ID_Context;Selected record number ([従業員]))
      End if
    End if
   \ (After)
     If (fCreation)
      $rc:=OD Insert in context (ID_Context)
     Else
      If (Selected record number ([従業員])=OD Number in context (ID_Context))
        $rc:=OD Update in context (ID_Context)
      End if
    End if
  End case
End if
```

このメソッドはコンテキストが有効かどうかチェックし、***OD Context state***関数を使ってアクティブにします。

レコードが登録されると、次の3ケースの中の1つが起こります：

- **レコードの作成処理：**レコードが作成されると、**Modified record**関数がBeforeフェーズでTrueを返します。登録されたレコードを有効にするには、Afterフェースの「fCreation」フラグを調べます。そして、サーバ上にレコードを作成するために***OD Insert in context***関数をコールします。
- **コンテキスト内で定義されたレコードの修正処理：**このメソッドは、コンテキスト内で定義されたレコードの数と選択レコード番号を比較して、コンテキストの一部であるレコードを調べます。そして、Beforeフェーズで***OD Goto in context***関数をコールして、正しいOracleの行にそのレコードを移動します。この行を更新するために、Afterフェーズで***OD Update in context***関数をコールします。
- **コンテキスト内で定義されていないレコードの修正処理：**4th Dimensionは、そのレコードを有効にします。

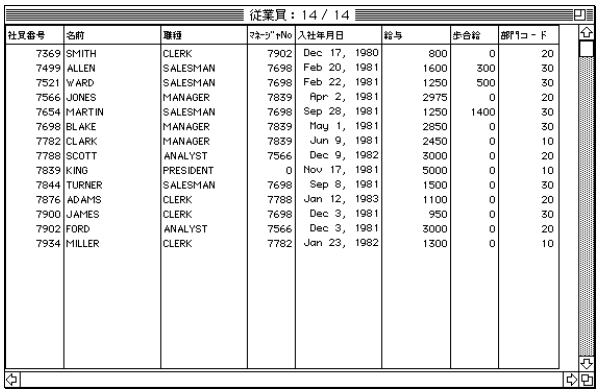
  このチュートリアルを簡単に行うために「ManageContext」メソッドは、カレントセレクション内の4th Dimensionレコードの順序を基にしてサーバ上で更新される行を調べるために選択レコード番号を使用します。ユーザ自身のアプリケーションの中には、データを更新するためにこの選択レコード番号を使用しないかもしれません。その代わり、それぞれの行に対して重複不可属性のプライマリキーを持つようにコンテキスト定義を構築する必要があるかもしれません。

コンテキスト使用によるレコードの作成と管理

「ManageContext」メソッドは、[従業員]テーブルのテーブルメソッドで呼び出されます。このテーブルメソッドを呼び出すには、次のように行います：

1. **「デザイン」モードに移動し、「メソッド」エディタを開き、[従業員]テーブルを選択して「開く」ボタンをクリックする。**
   テーブルメソッドを使用することにより、使用しているフォームに関係なく更新処理が行われることを保証することができます。

2. **「ユーザ」モードに移動し、「Activate」メソッドを実行する。**
   すると、次のような画面が表示されます。



3. **「SMITH」のレコードをダブルクリックする。**
   そのレコードが入力フォームの中に表示されます。

4. **SMITHの給与を1500ドルに変更して、「登録」ボタンをクリックする。**
   すると、対応しているOracleの行がサーバ上で更新されます。

5. **「更新」メニューから「新規レコード」を選択して、レコードを新しく作成する。**

6. **そのレコードに任意情報を入力して、「登録」ボタンをクリックする。**
   新規レコードがサーバ上に作成されます。

7. **「Deactivate」メソッドと「Activate」メソッドを実行して、サーバ上で行った変更を有効にする。**
   これにより、4th Dimensionレコードを出力フォームを通して削除し、サーバ上の更新されたレコード群をロードします。

   削除されるレコードが、ユーザにそのレコードを表示するために一時的にメモリ内に保持されるだけで決して4th Dimensionの中に保存されないことに注目することが重要です。データはOracleサーバ上に格納されるので、4th Dimensionのデータファイルにそのレコードを保存する必要はありません。

まとめ

まるで、Oracleサーバ上のデータが4th Dimensionのデータファイルから生じたかのように、Oracleサーバ上で行うデータ処理のアプリケーションをわずか数行程度のコードで簡単に作成することができます。この技法は、「ユーザ」モードと「ランタイム」モードにおける入力フォームで機能します。

もし、自分自身のアプリケーションでこの技法を使用する場合は、完璧なデータ修正システムを作成するために、いくつかの変更処理をメソッドで行なわなければならないことを覚えておいてください。前節では、4th DimensionのレコードとOracleの“SCOTT.EMP”テーブルのカラム間のリンクが、4th Dimensionの選択レコード番号を通して設定されました。しかし、もし、ソートや削除処理等によって選択順序が変更されると、「ManageContext」メソッドは、カレントのOracleのカラムを修正することができなくなります。また、Oracleサーバ上に新しく作成されたレコードを修正することもできなくなってしまいます。

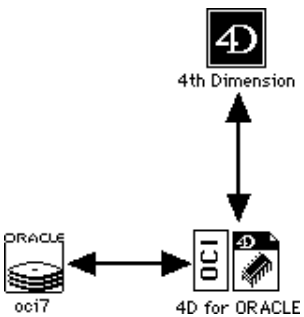
もし、信頼性の高いデータ修正システムを作成するには、Oracleのテーブルに対してプライマリキーを宣言する必要があります。コンテキストの定義にこの属性を指定することができます。

ローレベルコマンドは、SQL文を使って4th DimensionとOracleの間の通信を管理するコマンドです。4D for Oracleのローレベルコマンドを使用することにより、コンテキストコマンドを使って実行されるさまざまな処理を行うことができます。また、Oracleサーバから行をロードして、データの追加、修正等を行うことができます。

コンテキストコマンドと違って、ローレベルコマンドはユーザに有効なSQL問い合わせ(クエリー)の構築を要求します。例えば、次のステートメントは“ EMP ”テーブルの中に行を追加します。

err:=*OD Set SQL in cursor* (Cursor_ID ; "INSERT INTO EMP(EMPNO, ENAME) VALUES(John, Black)")

Oracleサーバ上でこのステートメントを実行するために、4D for OracleはOracleが理解できるステートメント内のローレベルコマンドを翻訳するためにOCI(Oracle Call Interfaces)を使用します。4D for Oracle内の各ローレベルコマンドごとに対応しているOCIがあります。



この章の終わりに4D for Oracleのローレベルコマンドとそれに対応したOCIコールの一覧表を掲載しています。

ローレベルコマンドの概要

ローレベルコマンドを使って、カーソル内のSQL文を実行します。「カーソル」とは、SQL文を参照したり制御するために使用されるメモリバッファのことです。各カーソルは、単一のSQL問い合わせ(クエリー)を制御します。同じ接続内で同時に複数のカーソルを作成したり、複数のステートメントを使用することができます。

ローレベルコマンドを使ったデータ管理

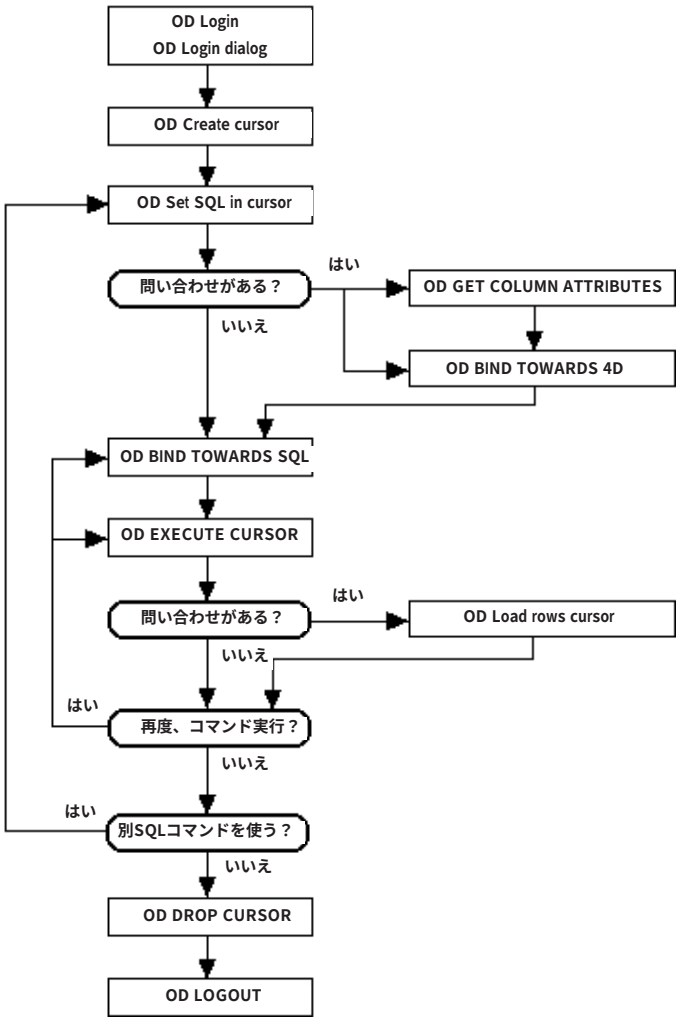
ローレベルコマンドを使って処理を制御するには、次のような基本ステップに従います：

1. ***OD Login*関数または*OD Login dialog*関数を使って、接続を作成する。**
   これらの関数は接続IDを返します。この接続IDはカーソルを作成する際に使用されます。
2. ***OD Create cursor*関数を使って、接続のカーソルを作成する。**
   この関数は、カーソルIDを返します。このカーソルIDは、この関数の後に続くカーソルを扱っているすべてのコマンドの中で使用されます。
3. ***OD Set SQL in cursor*関数を使って、SQL文を指定する。**
   SQL文を有効にするには、4th Dimensionの中にOracleのデータをロードしたり、または4th DimensionからOracleにデータを送信できるバインドを作成する必要があります。
   コンテキストメソッド内の双方向のバインドと違って、ローレベルメソッド内のバインドは1方向にしか機能しません。つまり、Oracleからの結果をロードするために1つのバインドを使用します。そして、Oracleにデータを送信するために別のバインドを使用します。

- SQL文でOracleの行を選択する場合は、***OD BIND TOWARDS 4D***コマンドを使って、その結果を4th Dimensionの中にロードできるバインドを作成します。
- SQL文でサーバ上の行を追加したり更新する場合は、***OD BIND TOWARDS SQL***コマンドを使って、4th Dimensionのデータを Oracleに送信できるバインドを作成します。

4. ***OD EXECUTE CURSOR*コマンドを使って、SQL文を実行する。**
5. **SQL文でOracleの行を選択したら、*OD Load rows cursor*関数を使ってその行をロードする。**
6. ***OD DROP CURSOR*コマンドを使って、カーソルを閉じる。**
7. ***OD LOGOUT*コマンドを使って、接続を閉じる。**

次の図は、SQLコマンドを実行するためのコール順序を示したものです。



カーソルの作成

カーソルIDを返す***OD Create cursor***関数を呼び出すことによりカーソルを作成することができます。カーソルIDは、この関数の後に続くコマンドの中でカーソルを確認するために使用されます。カーソルの使用が終了したら、***OD DROP CURSOR***コマンドを使ってカーソルを閉じます。

例えば、次のメソッドはサーバに接続し、2つのカーソルを作成し、それからそのカーソルを削除します。

```
ID_login:=OD Login dialog
ID_cursor1:=OD Create cursor (ID_login)
ID_cursor2:=OD Create cursor (ID_login)
OD DROP CURSOR (ID_cursor1)
OD DROP CURSOR (ID_cursor2)
```

カーソルの作成は、リソースがサーバレベルで割り当てられるため時間を要します。パフォーマンスを最適化するには、すべてのカーソルをまとめて作成し、アプリケーションが終了する際にすべてのカーソルを削除します。

サーバ上でオープンできる最大数の下でカーソルの数が保持されていることを覚えておいてください。この数は、Oracleサーバの「config.ora」ファイルによって決定されたサーバ構成に依存します。

注：*OD ROLLBACK*コマンドと*OD COMMIT*コマンドは、接続内の問い合わせ(クエリー)のすべてを有効にしたり取り消すことができます。しかし、これらのコマンドで個々のカーソルを有効にしたり取り消すことはできません。

サーバの接続を解除すると、接続内のすべてのカーソルが自動的に閉じられます。

データの選択

カーソル内にSELECT文を配置しそのカーソルを実行することにより、データを選択することができます。***OD Set SQL in cursor***関数を使用することにより、有効なSELECT文を指定することができます。

例えば、次のメソッドは「ENAME」カラム内のすべてのデータを選択するSELECT文を定義します。

```
$query:="SELECT ENAME FROM EMP"
$rc:=OD Set SQL in cursor (ID_cursor ; $query)
```

注：コードを読みやすくするSQL文を格納するために変数「$query」を使用します。そして、*OD Set SQL in cursor*関数に1つのパラメータ(引数)としてそのSQL文を受け渡すことができます。

選択データのロード

選択されたデータをロードするには、通常、次のようなステップを実行します：

1. **結果を取り込む4th Dimensionのオブジェクトを指定する。**
   結果をロードするオブジェクトを指定するには、***OD BIND TOWARDS 4D***コマンドを使って、各Oracleのカラムと4th Dimensionのフィールドや変数、配列をバインドします。
2. **カーソルを実行する。**
   カーソルを実行するには、***OD EXECUTE CURSOR***コマンドを使用します。
3. **結果の行をロードする。**
   OD Load rows cursorコマンドを使って行をロードします。そして、ロードしたい行の数を指定し、単一レコードまたはレコード群に結果を表示します。
   例えば、次のメソッドは“ ENAME ”カラム内のデータを選択するSELECT文を定義します。データをロードする前に、4th Dimensionのフィールドとそのカラムをバインドします。そして、カーソルを実行し、単一のカラムの結果をロードします。

   ```
   ID_Cursor:=OD Create cursor (vLogin)
   $query:="SELECT ENAME FROM EMP"
   $rc:=OD Set SQL in cursor (ID_cursor ; $query)
   OD BIND TOWARDS 4D (ID_cursor ; 1 ; >>[従業員]名前)
   OD EXECUTE CURSOR (ID_cursor)
   $rc:=OD Load rows cursor (ID_cursor ; 1)
   ```

   この例では、フィールドポインタがデータがロードされる4th Dimensionのオブジェクトを指し示すために使用されています。

オブジェクトの指定

これには、いくつかの選択肢があります。つまり、フィールドや変数、配列を指定したりまたはフィールドポインタや変数ポインタ、配列ポインタを指定することができます。
任意の変数とカラムをバインドして複数の結果行をロードすると、その変数の内容は最終の結果行と一致します。
任意の配列とカラムをバインドすると、4D for Oracleは配列の最後にその結果行を追加します。
4th Dimensionオブジェクトの指定に関する詳細は、後述の“ 4th Dimensionオブジェクトの指定 ”の節を参照してください。

サーバへのデータ送信

サーバにデータを送信するには、通常、次のようなステップを実行します：

1. **サーバ上でデータを追加、更新するSQL文を指定する。**
   次のような***OD Set SQL in cursor***ステートメントの中にこのSQL文を配置する。

   ```
   $query:="INSERT INTO EMP (EMPNO, ENAME) VALUES (123, 'Adams')"
   $rc:=OD Set SQL in cursor (ID_cursor ; $query)
   ```

   この例の問い合わせは、“ EMP ”テーブルの「EMPNO」と「ENAME」カラムの中に2つの値を追加します。この場合、追加されたデータ(123,'Adams')は、コンテキストとして指定されます。次の節では、データ元である4th Dimensionのフィールドや変数、配列を使って、さらに柔軟性のあるステートメントの構築方法を学習します。

2. **必要なら、Oracleサーバに送信されるデータ元である4th Dimensionのオブジェクトを指定する。**
   OD Set SQL in cursor関数で定義されたSQL文の中に4th Dimensionのオブジェクトを指定することができます。あるいは、***OD BIND TOWARDS SQL***コマンドを使ってそのデータ元のオブジェクトを指定することができます。

   OD BIND TOWARDS SQLコマンドを使用することにより、4th Dimensionのオブジェクトと Oracleのカラムをバインドすることができます。4th Dimensionのオブジェクトには、フィールドポインタや変数ポインタ、配列ポインタを指定することができます。
   4th Dimensionオブジェクトの指定に関する詳細は、後述の“ 4th Dimensionオブジェクトの指定 ”の節を参照してください。

3. **カーソルを実行する。**
   カーソルを実行するには、***OD EXECUTE CURSOR***コマンドを使用します。カーソルを実行すると、サーバ上の対応している行が更新されます。

4th Dimensionオブジェクトの指定

ここでは、ローレベルコマンド内の4th Dimensionオブジェクトの指定方法について説明します。この節では、まず最も簡単な指定方法を始めに紹介し、多種に渡る効果的な方法を徐々に紹介していきます。
指定されるオブジェクトはサーバに送られるデータ元、あるいはサーバからデータを取り込むために必要なオブジェクトです。

例えば、次のSQL文を見てみてください。

```
INSERT INTO EMP (EMPNO, ENAME) VALUES (123, 'Adams')
```

これはクエリーでないので、結果を復元する必要はありません。追加される値(123,'Adams')がコマンドテキストの中に組み込まれているので、この値の4th Dimensionオブジェクトを指定する必要はありません。4D for Oracleのコマンドシーケンスは、次のようになります：

```
ID_login:=OD Login dialog                    `またはOD loginコマンド
ID_cursor:=OD Create cursor (ID_login)
$query:="INSERT INTO EMP (EMPNO, ENAME) VALUES (123, 'Adams')"
$rc:=OD Set SQL in cursor (ID_cursor ; $query)
OD EXECUTE CURSOR (ID_cursor)
OD DROP CURSOR (ID_cursor)
OD LOGOUT (ID_login)
```

この例はとても簡単ではありますが、追加したい値が変更する度にSQL文を再定義しなければならないため、上記のコードは柔軟性に欠けています。これは、引数がたくさんある場合やダイアログボックスから情報を取得してくる際にとても邪魔になってしまいます。

代用変数の使用

直接、固定値を指定する代わりにSQL文の中で4th Dimensionの変数を使用することができます。例えば、次のメソッドは変数を使用して、サーバ上で追加されるデータを定義しています。

```
vEmpno:=123
vEname:="Adams"
$query:="INSERT INTO EMP (EMPNO, ENAME) VALUES (ｦvEmpnoﾈ, ｦEnameﾈ)"
$rc:=OD Set SQL in cursor (ID_cursor ; $query)
OD EXECUTE CURSOR (ID_cursor)
```

変数「vEmpno」と「vEname」は、固定値“ 123 ”と“ Adams ”を置き換えます。これらの変数は、「代用変数」と呼ばれます。SQL文の中の代用変数は、“ｦ”と“ﾈ”で囲む必要があります。

この代用変数には、次のような2つの利点があります：

- ユーザがSQLコマンドの引数を知る前にSQLコマンドを構築できる
- 代用変数を使っているSQLコマンドは、値が変更された際に**OD Set SQL in cursor**関数によって解析し直す必要がない

3人の従業員(123,'Adams')、(124,'Adamo')、(125,'Adam')を追加するには、次のようなメソッドを使用します：

```
vEmpno:=123
vEname:="Adams"
$query:="INSERT INTO EMP (EMPNO, ENAME) VALUES (ｦvEmpnoﾈ, ｦvEnameﾈ)"
$rc:=OD Set SQL in cursor (ID_cursor ; $query)
OD EXECUTE CURSOR (ID_cursor)
vEmpno:=124
vEname:="Adamo"
OD EXECUTE CURSOR (ID_cursor)
vEmpno:=125
vEname:="Adam"
OD EXECUTE CURSOR (ID_cursor)
```

入力データをユーザに尋ねるダイアログボックスを使って、変数「vEmpno」と「vEname」に代入する行を置き換えることができます。

もし、データベースをコンパイルする予定があるなら、**C_INTEGER**や**C_INTEGER**のようなコンパイラ命令を使って代用変数のタイプを宣言しておく必要があります。

接頭辞に“ $ ”の付くローカル変数を代用変数として使用することはできません。

代用フィールドの使用

直接、固定値を指定する代わりにSQL文の中で4th Dimensionのフィールドを使用することができます。代用フィールドは、“ ﾇ ”と“ ﾈ ”で囲んで、“ ﾇ[従業員]名前ﾈ ”のような書式で指定する必要があります。

例えば、[従業員]テーブルに「社員番号」フィールドと「名前」フィールドがあると仮定します。次のようなメソッドを使って、“ EMP ”テーブルの中に[従業員]テーブルのカレントセレクションの値を追加することができます：

```
$query:="INSERT INTO  EMP (EMPNO, ENAME) VALUES (ﾇ[従業員]社員番号ﾈ, ﾇ[従業員]名前ﾈ)"
$rc:=OD Set SQL in cursor (ID_cursor ; $query)
APPLY TO SELECTION ([従業員] ; OD EXECUTE CURSOR (ID_cursor))
```

4D for Oracleは、***OD EXECUTE CURSOR***コマンド実行時のカレントレコードのフィールド値を使用します。もし、カレントレコードがない場合は、***OD EXECUTE CURSOR***コマンドは何も行いません。

代用配列の使用

直接、固定値を指定する代わりにSQL文の中で4th Dimensionのフィールドを使用することができます。配列を使用すると、4D for Oracleは配列の中にある要素数と同じ数のSQL文を実行します。代用変数や代用フィールドと同じように、SQL文の中の代用配列は“ ﾇ ”と“ ﾈ ”で囲む必要があります。

例えば、次のメソッドは配列「tEmpno」と「tEname」の中の値を基にサーバ上に3つの行を追加します：

```
ARRAY STRING (30;tEmpno;3)
ARRAY INTEGER (tEname;3)
$query:="INSERT INTO EMP (EMPNO, ENAME) VALUES (ﾇtEmpnoﾈ, ﾇtEnameﾈ)"
$rc:=OD Set SQL in cursor (ID_cursor ; $query)
tEname{1}:="ADAMS"
tEname{2}:="ADAMO"
tEname{3}:="ADAM"
tEmpno{1}:=123
tEmpno{2}:=124
tEmpno{3}:=125
OD EXECUTE CURSOR (ID_cursor)
```

配列使用における注意点

- 変数やフィールドが配列で使用されている場合、***OD EXECUTE CURSOR***コマンドは、その配列の最初の行を使用することにより、たった1度しか実行されません。
- 複数の配列が代用変数として使用されている場合、それらは同じ行数を持っている必要があります。

フィールドポインタ、変数ポインタ、および配列ポインタの使用

SQL文の文字列の中にフィールド名や変数名、配列名を直接組み込む代わりに、代用のポインタを使用することができます。

ポインタを使用するには、代用のフィールドや変数、配列を***OD Set SQL in cursor***ステートメント内で空(ヌ ネ) のままにしておき、ステートメントを構築する際にそれのポインタを組み込みます。***OD BIND TOWARDS SQL***コマンドと***OD BIND TOWARDS 4D***コマンドは変数や配列、フィールドのポインタを受け入れます。

次のメソッドは、サーバ上の2つの行に2つの変数内の値を追加します。変数のポインタは、***OD BIND TOWARDS SQL***ステートメントの中で指定されます。

```
vEmpno:=123
vEname:="Adams"
$query:="INSERT INTO EMP (EMPNO, ENAME) VALUES (ヌネ, ヌネ)"
$rc:=OD Set SQL in cursor (ID_cursor ; $query)
OD BIND TOWARDS SQL (ID_cursor ; 1 ; >>vEmpno)
OD BIND TOWARDS SQL (ID_cursor ; 2 ; >>vEname)
OD EXECUTE CURSOR (ID_cursor)
```

ポインタ使用における利点

フィールドや変数、配列を再定義しなければならない度にSQL文を再構築する必要がありません。例えば、フィールド名やそのフィールドの属するテーブルの名前を変更しても、ポインタはそのまま有効なので、そのフィールドを参照するすべてのSQL文の中の修正箇所を複製する必要がありません。
4th Dimensionのクロスリファレンス生成ツールである4D Insiderが、ポインタによってそれらが参照される場合にしかメソッド内で使用されるフィールドや変数、配列を認識しなく済みます。

OCI — 4D for Oracleの同等コマンド

OCI(Oracle Call Interfaces)内のすべてのコールは、4D for Oracleからアクセスされます。OCAN—***OD CANCEL LOADING***のようなコールは、トランザクションなしに4th Dimension用に適合されるようになりました。また、その他のコールも機能が拡張され、コマンド内の参照データの追加や自動データ変換等をサポートできるようになりました。

OCIは、4D for Oracleおよび「oci7」Oracleドライバ内で実行されます。

次の表は、4D for Oracleコマンドや関数と同じ意味を持つOCIコールを一覧にしたものです：

OCIコール	4D for Oracleコール
OBNDRN	*OD BIND TOWARDS SQL*
OBNDRV	*OD BIND TOWARDS SQL*
OCAN	*OD CANCEL LOADING*
OCLOSE	*OD DROP CURSOR*
OCOF	*OD SET OPTIONS*
OCOM	*OD COMMIT*
OCON	*OD SET OPTIONS*
ODEFIN	*OD BIND TOWARDS 4D*
ODSC	*OD GET COLUMN ATTRIBUTES*
OERMSG	*OD Last error*
OEXEC	*OD EXECUTE CURSOR*
OEXN	*OD EXECUTE CURSOR*
OFEN	*OD Load rows cursor*
OFETCH	*OD Load rows cursor*
OLOGOF	*OD LOGOUT*
OLON	*OD Login*
ONAME	*OD Get column title*
OOPEN	*OD Create cursor*
ORLON	*OD Login*
OROL	*OD ROLLBACK*
OSQL3	*OD Set SQL in cursor*

上記のコマンドや関数に加えて、4D for Oracleは次の2つの関数を新しく追加しました。***OD Number rows processed***関数は、ロードや更新、削除される行の数を調べることができます。***OD Number of columns***関数は、ロードされるカラムの数を返します。

この章にあるコマンドは、Oracleサーバへのログインとログアウトを可能にして接続におけるいくつかのハイレベルの操作を実行します。

ログインした際に、4D for Oracleは、コマンドが適用される接続を識別するのに多くの4D for Oracleコマンドで使われる接続IDを返します。また、接続IDはサーバからログアウトする際にも使われます。

ログインコマンドとログアウトコマンド

この節のコマンドにより、Oracleサーバへの接続を制御することができます。ログインコマンドを使って、次の事柄を行なうことができます：

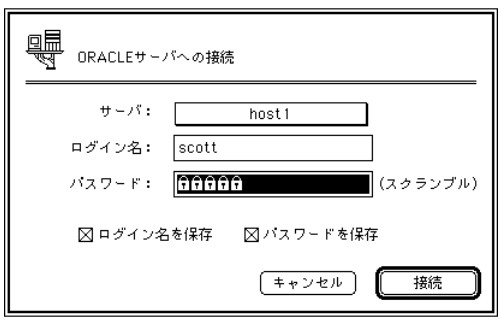
- ダイアログボックスを使ってログインする（***OD Login dialog***）
- ログイン引数を指定してログインする（***OD Login***）
- ログアウトする（***OD LOGOUT***）
- 特定の接続が存在しているかを調べて、ログイン時に指定されたログイン名とサーバを取得する（***OD Login state***）

OD Login dialog

OD Login dialog **({モード})→ 倍長整数**

引数	タイプ	説明
モード	整数	モード選択

OD Login dialog 関数は、「Oracle サーバへの接続」ダイアログボックスを表示して、ユーザがサーバを選択しログイン名とパスワードを入力することを可能にします。



ユーザが「接続」ボタンをクリックすると、***OD Login dialog*** 関数はログインして接続IDを返します。ユーザが「キャンセル」ボタンをクリックすると、0を返します。エラーが発生した場合には、-1を返します。

ユーザは「サーバ」ポップアップメニューからサーバを選択します。ポップアップメニューは、ユーザのMacintosh上へのSQL*Netのインストール時に作成されたconfig.oraファイルで定義されたサーバを一覧表示します。[Oracle Home」フォルダに格納されたconfig.oraファイルは、各サーバのエイリアスを含んでいます。SQL*Net 1.5では、config.oraファイルの各行は、サーバを定義して次のフォーマットで指定されなければなりません：

エイリアス名 = ネットワークアドレス

例えば、config.oraファイルからの次の文は、Marketing AppleTalkエリアの Macintoshコンピュータ上にあるClientsというサーバと、アドレス192.9.200.8のTCP/IP下のUnixコンピュータ上にあるMISというサーバを定義します。

```
MyClients = AT:Clients,Marketing
BigBrother = T:192.9.200.8:MIS
```

この場合、「サーバ」ポップアップメニューは、前に定義された「Oracle for Macintosh」サーバおよびMyClientsとBigBrotherサーバを表示します。「Oracle for Macintosh」は、Macintosh上のシングルユーザ版のOracleサーバへのログインを可能にします。

バージョン2のSQL*Netを使用している場合は、「Oracleサーバへの接続」ダイアログボックス内にサーバのエイリアスリストを表示するために（例えば、SimpleText等を使って）“ config.ora ” ファイルを作成します。

引数「モード」が0の場合、4D for Oracleはバージョン6のOracleとして動作します。

「モード」が1の場合、ユーザが接続したバージョン(バージョン6のOracleまたはOracle7)でデータベースはいつものように動作します。

「モード」が2の場合、4D for OracleはOracle7として動作します。もし、バージョン6のOracleデータベースに接続してる際にこのモードを使用すると、1番目の問い合わせが実行された時にエラーORA-1011の “ Cannot use this language type when talking to V6 database ” というエラーメッセージが表示されます。

バージョン6の動作では、CHARカラムは可変長です(CREATE TABLE文によって作成されるカラムを含む)。Oracle7の動作では、CHARカラムは固定長です。

また、バージョン6では、固定長文字列は内部データタイプ1を持っています。Oracle7では、固定長文字列は内部データタイプ96を持っています。

4th Dimensionは、ユーザによって選択されたサーバの名前をリソースに保存します。また、「ログイン名を保存」チェックボックスが選択された場合には、ログイン名が保存され、「パスワードを保存」チェックボックスが選択された場合には、ユーザのパスワードが保存されます。

警告：データ保護が重要な際には、ユーザがパスワードを保存しないことをお薦めします。ユーザがパスワードを保存した場合には、ユーザの4th Dimensionデータベースへのアクセスで誰でもサーバに接続することができます。さらに、パスワードは保存されている間、暗号化されません。

OD Login dialog 関数の呼び出し後、ユーザがログイン時に何を選択したかを調べるのに***OD Login state*** 関数を使うことができます。

注：ダイアログボックスでユーザが「接続」ボタンをクリックした場合には、OKシステム変数が1に設定されます。ユーザが「キャンセル」キャンセルをクリックした場合には、0に設定されます。

参照：OD Login、OD LOGOUT、OD Login state

OD Login

OD Login **({ログイン} ; {パスワード} ; {サーバ} ; {モード}) → 倍長整数**

引数	タイプ	説明
ログイン	文字列	ログインの名前
パスワード	文字列	パスワード
サーバ	文字列	サーバの名前
モード	倍長整数	モード選択

OD Login 関数は、指定したログイン引数を使ってOracleサーバにログインして接続IDを返します。

「ログイン」は、使用したいログインの名前です。「ログイン」を指定しない場合には、4D for Oracleは「Oracle サーバへの接続」ダイアログボックスにおいて前回に入力されたログイン名を使います。ログイン名は、ダイアログボックスで「ログイン名を保存」チェックボクックスが選択された場合にのみ認識されています。

「パスワード」は、「ユーザ」に対するパスワードです。「パスワード」を指定しない場合には、4D for Oracleは「Oracle サーバへの接続」ダイアログボックスにおいて前回に入力されたパスワードを使います。「パスワード」は、ダイアログボックスで「パスワードを保存」チェックボックスが選択された場合にのみ使われます。

「サーバ」は接続したいサーバを指定します。Oracle TNSNAMES.ORAファイルで定義されたエイリアス、完全なネットワークアドレス、またはシングル-ユーザバージョンのOracleサーバへログインするのに使われる " Personal Oracle 7 " 文字列を使うことが可能です。

「サーバ」を指定しない場合には、4D for Oracleは「Oracle サーバへの接続」ダイアログボックスで選択された最後のサーバを使います。

「モード」が0の場合、4D for Oracleはバージョン6のOracleとして動作します。

「モード」が1の場合、ユーザが接続したバージョン(バージョン6のOracleまたはOracle7)でデータベースはいつものように動作します。

「モード」が2の場合、4D for OracleはOracle7として動作します。もし、バージョン6のOracleデータベースに接続してる際にこのモードを使用すると、1番目の問い合わせが実行された時にエラーORA-1011の " Cannot use this language type when talking to V6 database " というエラーメッセージが表示されます。

バージョン6の動作では、CHARカラムは可変長です(CREATE TABLE文によって作成されるカラムを含む)。Oracle7の動作では、CHARカラムは固定長です。

また、バージョン6では、固定長文字列は内部データタイプ1を持っています。Oracle7では、固定長文字列は内部データタイプ96を持っています。

▼ 例の例は、ログイン名 " scott " とパスワード " tiger " を使って、Oracle7の " MyClients " サーバにログインします：

```
Login_ID := OD Login  ("scott" ; "tiger" ; "MyClients" ; 2)
```

▼ 次の例は、「Oracle サーバへの接続」ダイアログボックスで最後に選択されたOracleのバージョン、サーバ、ログイン名、パスワードを使ってログインします。

```
Login_ID := OD Login
```

OD Login 関数は、操作が成功した場合には接続IDを返し、エラーがある場合には-1を返します。上記の文では、接続IDは変数「Login_ID」に格納されます。そして、この変数はログインに関する情報を取得するのに***OD Login state*** 関数と一緒に使用されます。また、サーバからログアウトする場合にも***OD LOGOUT*** コマンドと一緒に使用されます。

参照：OD Login dialog、OD LOGOUT

OD LOGOUT

OD LOGOUT **(ログインID)**

引数	タイプ	説明
ログインID	倍長整数	接続ID

OD LOGOUT コマンドは、指定した接続を終了します。

「ログインID」は、***OD Login*** 関数または***OD Login dialog*** 関数を使ってログイン時に取得された有効な接続IDです。

接続時にオープンされたカーソルとコンテキストはクローズされて、「ログインID」識別子は無効になります。しかし、「ログインID」は、自動的にゼロにリセットされないので「ログインID」の値を接続がオープンされているかどうかを調べるのに使うことはできません。接続が現在オープンされているかどうかを調べるには、***OD Login state*** 関数を使います。

ログアウト時に、4D for Oracleはカレントトランザクションをコミットします。

4th Dimensionを終了した際に、4D for Oracleは自動的にサーバからログアウトします。

参照：OD Login、OD Login dialog

OD Login state

OD Login state **(ログインID ; {ログイン} ; {サーバ}) → 整数**

引数	タイプ	説明
ログインID	倍長整数	接続ID
ログイン	文字列	ログインの名前
サーバ	文字列	サーバID

OD Login state 関数は、「ログインID」がオープン接続を参照しているかどうかを示す整数を返します。

「ログインID」がオープン接続を識別した場合には、***OD Login state*** 関数は1を返してユーザの名前を「{ログイン}」変数に、サーバIDを「{サーバ}」変数に格納します。サーバIDは、ログインで使われた方法により、Oracle config.oraファイルからのサーバのエイリアスまたはサーバのネットワークアドレスのいずれかとなります。

「Oracle サーバへの接続」ダイアログボックスを使ってログインした場合には、返されたサーバIDは、config.oraファイルからのサーバのエイリアスです。***OD Login*** 関数を使ってログイン引数を指定した場合には、返されたサーバIDは、config.oraファイルからのエイリアスまたは完全なネットワークアドレスのいずれかです。

「ログインID」がオープン接続を識別しなかった場合には、***OD Login state*** 関数は0を返して「サーバ」と「ユーザ」変数の値を変更しません。

▼ 例えば、次のメソッドはユーザにログインのダイアログボックスを表示して正しいサーバにログインしたかどうかを調べます：

```
Login_ID := OD Login dialog
If (OD Login state (Login_ID ; User ; Server) = 1)
   If (Server # "BigBrother")
      ALERT ("BigBrotherにのみ接続します！")
   ELSE
      ALERT (User + "さん！" + Char (13) + "ようこそ" + Server + "へ")
   End if
ELSE
   ALERT ("ログインできませんでした。")
End if
```

参照：OD Login、OD Login dialog

ハイレベルコマンド

いくつかのハイレベルコマンドは、接続段階での操作を実行することができます。これらのコマンドは1つの簡単なコマンドを使って複合関数を実行できるローレベルのAPIコマンドと連鎖しています。これらのコマンドを使って、次の事柄を行なうことができます：

- SQL文を実行する（***OD Execute SQL***）
- Oracleサーバ上に4th Dimensionテーブルのクローンを作成する（***OD Clone 4D table***）
- トランザクションをコミットまたはロールバックする（***OD COMMIT***、***OD ROLLBACK***）

OD Execute SQL

OD Execute SQL **(ログインID ; SQLコマンド ; 制限; {ポインタ1;...; ポインタ22}) → 整数**

引数	タイプ	説明
ログインID	倍長整数	接続IDまたはカーソルID
SQLコマンド	テキスト	実行するSQLコマンド
制限	倍長整数	返される最大行数
ポインタN	ポインタ	4th Dimensionフィールド、変数、または配列のポインタ

OD Execute SQL 関数は引数「ログインID」を指定することによりSQL問い合わせを送って、結果を4th Dimensionフィールド、変数、または配列に格納することができます。

また、Oracleは各問い合わせに対してカーソルが必要なので、4D for Oracleは内部カーソルのセットを管理します。これにより、***OD Execute SQL*** 関数はユーザ自身のカーソルを使用するので、***OD Execute SQL*** 関数を複数呼び出した際に内部カーソル管理のオーバーヘッドを減らします。

「ログインID」は、有効な接続IDまたはカーソルIDでなければなりません。

「SQLコマンド」は、実行するSQL問い合わせのテキストです。すべてのSQLコマンドは、シンタックスが有効な限り受け付けられます。Oracleによってサポートされ ているSQLに関する詳細は、『SQL*ランゲージリファレンスマニュアル』を参照してください。

「制限」は、4th Dimensionに返される結果行の最大数を指定します。「制限」が-1である場合には、***OD Execute SQL*** 関数はすべての結果をロードします。-1以外の数が指定された場合には、残りの行はロードされません。制限を設定して後で残りの行をロードするには、第7章の“ローレベルコマンド”で説明されているカーソルコマンドを使います。

「ポインタ1;...; ポインタ22」は、問い合わせの結果を受け取るフィールド、変数、または配列のポインタです。「ポインタN」によって参照されるオブジェクトは、存在すれば、カラムNからの結果を取り出します。

OD Execute SQL 関数は、結果のカラム数またはエラーが発生した場合には-1を返します。

配列処理

4th Dimensionの配列を使って処理を行うと、4D for Oracleはパフォーマンスを上げる配列処理（いくつかの行ブロックを使ってデータの送受信を行う処理）を利用します。

Oracleデータベースに4th Dimensionのカレントセレクションを書き出すには、次のようなメソッドを作成します：

ARRAY TO SELECTION (**[従業員];**rc:=***OD Execute SQL*** (login ; "INSERT INTO EMP (ENAME) VALUES (ﾇ[従業員]名前)ﾈ)"))

この方法の利点は、配列処理を使用していないにもかかわらず、それがわずか1行で構成されているところです。従って、4th Dimensionのレコードが1行ずつOracleに送られるので実行タイムを短縮できます。

配列処理を利用するには、次のようなメソッドを作成します：

SELECTION TO ARRAY ([従業員]名前 ; tName)
rc:=***OD Execute SQL*** (login ; "SELECT INTO EMP (ENAME) VALUES (ﾇtNameﾈ)")
CLEAR VARIABLE (tName)

この方法は4th Dimensionの配列を介して使用してるため、1番目の方法よりもメモリをたくさん必要とします。そのため、4th Dimensionおよび4D Clientのアプリケーションに十分なメモリを割り当てる必要があります。そうしないと、レコードのセレクションがばらばらに分かれて書き出されてしまいます。

4D for Oracleは、次のルーチンの1つを実行することにより、配列処理を使用します：

- ***OD Execute SQL***
- ***OD EXECUTE CURSOR***
- ***OD Load rows cursor***
- ***OD Load rows context***

問い合わせを送信するためのカーソルの使用

次の例は、***OD Execute SQL*** 関数を使って2つの問い合わせを送るために同じカーソルを使用します：

Login_ID:=***OD Login dialog***
Cursor_ID:=***OD Create cursor*** (Login_ID)
$rc:=***OD Execute SQL*** (Cursor_ID ; "DELETE FROM EMP")
$rc:=***OD Execute SQL*** (Cursor_ID ; "INSERT INTO EMP(EMPNO, ENAME") VALUES (42, 'johns')")
OD DROP CURSOR (Cursor_ID)
OD LOGOUT (Cursor_ID)

結果行の抽出

空の問い合わせと一緒に***OD Execute SQL*** 関数にカーソルIDを指定すると、4D for Oracleはその問い合わせを再実行することなく残りの行をロードします。これにより、次の例のように1行1行結果を取り込むことができます。

▼ これは***OD Execute SQL*** 関数を使ってSELECT文を実行し、10行の結果を取り込みます：

```
Login_ID:=OD Login dialog
Cursor_ID:=OD Create cursor (Login_ID)
  `任意行を取り込むことなく問い合わせを送る
$rc:=OD Execute SQL (Cursor_ID ; "SELECT ENAME FROM EMP" ; 0 ; >>tEname)
Repeat
  `次の10行を取り込む
   $rc:=OD Execute SQL (Cursor_ID ; "" ; 10)
  `取り込む行がなくなるまで、
Until (OD Cursor state (Cursor_ID) = 2)
OD DROP CURSOR (Cursor_ID)
OD LOGOUT (Login_ID)
```

注：

- 残り行を取り込むために問い合わせとしてヌルストリングを受け渡すと、定義されたポインタタイプの引数は無視されます。結果を取り込む4th Dimensionオブジェクトは、問い合わせのテキストを送信することによって***OD Execute SQL*** 関数を呼び出すと指定されます。
- カーソルに適用されるローレベルコマンドは、***OD Execute SQL*** 関数によって使用されているカーソルをそのまま利用することができます。上記の例では、読み込まれる結果の最後をユーザがわかるかどうかを調べるために***OD Cursor state*** 関数を使用しています。

参照：OD Login dialog、OD Create cursor、OD Cursor state

「ポインタN」引数

問い合わせが結果を返した場合、または「SQLコマンド」がSELECT...タイプであった場合には、***OD Execute SQL*** 関数は「ポインタN」が指すオブジェクトのタイプに応じて次のように動作します：

- 「ポインタN」がフィールドを指す場合には、***OD Execute SQL*** 関数はロードされている行と同じ数のレコードを作成します。「制限」が1である場合には、メモリにレコードを作成して結果行を取り出しますが保存しません。4th Dimensionに結果を保持するかを決定する**SAVE RECORD**コマンドは呼び出されません。結果が4th Dimensionレコードにロードされた際には、レコードのカレントセレクションは、4D for Oracleによって作成されたレコードに設定されます。
- 「ポインタN」が変数を指す場合には、返された最後の結果行のカラムNの値を受け取ります。変数は一度に1行だけを保持できるということを覚えておいてください。
- 「ポインタN」が配列を指す場合には、配列はそれぞれのカラムの値を受け取ります。

OD Execute SQL 関数は、結果を変換して、その結果は「ポインタN」によって参照されるオブジェクトのタイプに一致します。「ポインタN」引数の数が返されるカラム結果の数に一致する必要はありません。余分な引数とカラムは無視されます。

問い合わせの実行

OD Execute SQL 関数は、SQL問い合わせを実行するのに最も簡単な方法です。この関数は、次の一連のステップと等しくなります：

1. ***OD Create cursor*** **関数を使って、カーソルを作成する。**
2. ***OD Set SQL in cursor*** **関数を使って、SQL問い合わせを送る。**
3. ***OD BIND TOWARDS 4D*****コマンドを使って、結果を取り戻すためにリンクを作成する。**
4. ***OD EXECUTE CURSOR*****コマンドを使って、問い合わせを実行する。**
5. ***OD Load rows cursor*** **関数を使って、結果をロードする。**
6. ***OD DROP CURSOR*****コマンドを使って、カーソルを削除する。**

Oracleサーバへのログイン

次のメソッドは、Oracleサーバにログインして、DEPTテーブルに行を追加し、ログアウトします。

```
Login_ID := OD Login dialog
If (Login_ID>0)
   $col := OD Execute SQL (Login_ID ; "INSERT INTO DEPT VALUES (50,
   'PRODUCTS', 'TOKYO' ; -1)
   OD LOGOUT (Login_ID)
End if
```

データのロード

次のメソッドは、EMPテーブルから[従業員]という4th Dimensionテーブルにデータをロードします：

```
$col := OD Execute SQL (Login_ID ; "SELECT EMPNO, ENAME, SAL FROM EMP";
            -1; >>[従業員]社員番号 ; >>[従業員]名前 ; >>[従業員]給与)
```

参照：OD Login、OD Login dialog、OD Cursor state

OD Clone 4D Table

OD Clone 4D Table **(ログインID ; {4Dテーブル番号} ; {オプション}) → 整数**

引数	タイプ	説明
ログインID	倍長整数	接続ID
4Dテーブル番号	整数	4Dのテーブル番号
オプション	倍長整数	オプション

OD Clone 4D Table 関数は、4th Dimensionテーブルに等しいカラム定義を使ってOracleのテーブルを作成します。

「ログインID」は、有効な接続IDでなければなりません。

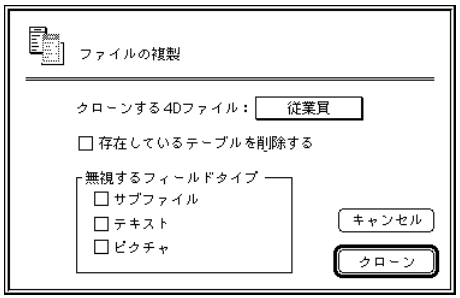
「4Dテーブル番号」は、4th Dimensionテーブルの番号で、複製されるストラクチャです。

「オプション」は、オプションの組み合わせです。値を加算することによって複数のオプションを選択できます。

オプションは、次の通りです：

kReplace=1	***OD Clone 4D Table*** 関数は新規テーブルを作成する前にDROP TABLE文を生成します。
kWithoutSubtable=2	***OD Clone 4D Table*** 関数はサブテーブルを無視します。サブテーブルのクローンを作成することは現在サポートされていません。このオプションは、サブテーブルを含んでいるすべてのテーブルに対して要求されなければなりません。
kWithoutText=4	***OD Clone 4D Table*** 関数は、タイプ「テキスト」のフィールドを無視します。
kWithoutPicture=8	***OD Clone 4D Table*** 関数は、タイプ「ピクチャ」のフィールドを無視します。

「4Dテーブル番号」が指定されていない場合には、***OD Clone 4D Table*** 関数は「テーブルの複製」ダイアログボックスを表示します。



ポップアップメニューはクローンされる4th Dimensionテーブルを選択するのを可能にして、チェックボックスは使用可能なオプションを表します。

OD Clone file 関数は、操作が成功した場合には1を返し、エラーが発生した場合には-1を、またはユーザが「キャンセル」ボタンをクリックした場合には0を返します。ユーザが「OK」ボタンをクリックした場合には、OKシステム変数は1に設定されます。「キャンセル」ボタンをクリックした場合には、OKシステム変数は0に設定されます。

新規テーブルは4th Dimensionテーブル名を持ち、そのカラムは4th Dimensionテーブルのフィールド名を持ちます。フィールド名がスペースを含んでいる場合には、スペースはアンダーラインに置き替えられます。

タイプは次の規則に従って変換されます：

4th Dimension	**Oracle**
文字(長さ)	CHAR(long)
テキスト	LONG
実数	FLOAT
整数	INTEGER
倍長整数	INTEGER
日付	DATE
時間	DATE
ブール	NUMBER(1)
ピクチャ	LONG RAW

4th Dimensionフィールドが「必須入力」属性を持っている場合には、対応するカラムはNOT NULLオプションを使って作成されます。

インデックスフィールドに対しては索引が作成されます。

索引の名前は“I_”を前に付けた4th Dimensionテーブル名、“$”記号、そしてフィールド名の連結したものです。例えば、EMPテーブルのSALフィールドに対する索引は、“I_EMP$SAL”と名付けられます。

4th Dimensionフィールドが「重複不可」属性を持っている場合には、索引はUNIQUEオプションで作成されます。

注：バージョン6のOracleでは、カラムの入力に対していくつかの制限があります。例えば、各テーブルごとに複数のLONG RAWタイプのカラムは存在しなくて、カラムの最大サイズは64Kです。また、Oracle7でも、カラムの最大サイズは64Kです。詳細は、『Oracleリファレンスマニュアル』を参照してください。

▼ 次のメソッドは、4th Dimensionデータベースにあるテーブルを Oracleサーバにコピーします。既にサーバ上にテーブルが存在する場合には、自動的に置き替えられます。

```
Login_ID := OD Login dialog
If (Login_ID>0)
  For ($i ; 1 ; Count files)
    rc := OD Clone file (Login_ID ; $i ; 1)
  End for
End if
```

参照：OD Login

OD COMMIT

OD COMMIT (ログインID)

引数	タイプ	説明
ログインID	倍長整数	接続ID

OD COMMITコマンドは、接続「ログインID」に対してSQL COMMITを送ります。

「ログインID」は、有効な接続IDでなければなりません。

すべての未処理のコマンドは、「ログインID」を介して送られます。Oracleサーバ上のすべてのデータの変更は、コマンドがコンテキスト、カーソル、または***OD Execute SQL***関数への呼び出しの構成のなかで実行されたかどうかに関係なく実行（INSERT、UPDATE、またはDELETE）されます。

このコマンドは、カレントトランザクションで実行されたすべての変更を恒久的にします。データはこの次の更新によって変更することが可能です。トランザクションをコミットすることは、トランザクションにあるすべてのセーブポイントを消去し、トランザクションを終了して、トランザクションのロックを解除します。

2つの独立したトランザクションが必要な場合には、それぞれに対して別の接続をオープンしなければなりません。

参照：OD ROLLBACK

OD ROLLBACK

OD ROLLBACK (ログインID)

引数	タイプ	説明
ログインID	倍長整数	接続ID

OD ROLLBACKコマンドは、「ログインID」を使って接続に対してSQL ROLLBACKコマンドを送ります。

「ログインID」は、有効な接続IDでなければなりません。

Oracleサーバ上のデータに変更をもたらすINSERT、UPDATE、またはDELETEのようなすべての未処理のコマンドは、コンテキスト、カーソル、***OD Execute SQL*** 関数の構成のなかで行なわれるかどうかには関係なくキャンセルされます。

参照：OD COMMIT

OD Clone Table

OD Clone Table (ログインID) → 整数

引数	タイプ	説明
テーブルID	倍長整数	ログインID

OD Clone Table 関数は、Oracleテーブルに等しいカラム定義を使って4th Dimensionのテーブルを作成します。

「ログインID」は、有効なログインIDでなければなりません。

OD Clone Table 関数を実行すると、次のような「テーブルの複製」ダイアログボックスが表示されます。

OD Clone table 関数は、操作が成功した場合には1を返し、エラーが発生した場合には-1を、またはユーザが「キャンセル」ボタンをクリックした場合には0を返します。

新規テーブルは“ テーブルN ”の名前を持ち、“ N ”はテーブル番号を示します。フィールド名は、Oracleデータベース上のカラム名です。

タイプは、次の規則に従って変換されます：

Oracle	**4th Dimension**
CHAR	**文字**
VARCHAR2	**文字**
DATE	**文字**
RAW	**文字**
ROWID	**文字**
NUMBER	**実数**
LONG	**テキスト**
LONG RAW	**ピクチャ**

コンテキストは、Oracleのテーブルにあるカラムまたは式と、4th Dimensionのフィールド、変数、あるいは配列との間のバインドから構成されます。一旦定義されると、コンテキストは、Oracleのデータを4th Dimensionにロードして SQL文を使わなくてもOracleサーバ上の情報を更新することができます。
この章のコンテキストコマンドで次の事柄を行なうことができます：

- コンテキストを作成する（***OD Create context dialog***、***OD Create context***）
- コンテキストでバインドを定義する（***OD ADD TO CONTEXT***）
- コンテキストに句を追加する（***OD EDIT CLAUSES IN CONTEXT***、***OD SET CLAUSE IN CONTEXT***）
- コンテキスト定義を保存またはロードする（***OD Save context picture***、***OD SAVE CONTEXT FILE***、***OD Open context picture***、***OD Open context file***）
- コンテキストをアクティブにする（***OD Activate context***）
- コンテキスト行を使った4Dセレクションの同時性（***OD Find in context***）
- Oracleの行を4th Dimensionにロードする（***OD Next in context***、***OD Previous in context***、***OD Goto in context***、***OD Load rows context***）
- データを更新、追加、削除する（***OD Update in context***、***OD Insert in context***、***OD Delete in context***）
- コンテキストに関する情報を取り出す（***OD Context state***、***OD Records in context***、***OD Number in context***、***OD Get clause in context***）
- コンテキストを非アクティブするか、またはクローズする（***OD DEACTIVATE CONTEXT***、***OD DROP CONTEXT***）

ダイアログボックスを使用したりまたはメソッドを実行することにより、コンテキストを作成することができます。これに関する詳細は、第3章の“コンテキストの使用”を参照してください。

複数のテーブルと関係しているコンテキスト、あるいはGROUP BY句またはDISTINCT句を含んでいるコンテキストはブラウズのみが可能です。この場合、Oracleテーブルのデータを更新するのにデータ変更コマンドである***OD Insert in context***、***OD Update in context***、***OD Delete in context*** を使うことはできません。

複数のテーブルと関係しているコンテキスト、あるいはGROUP BY句を含んでいるコンテキストは、行から行へ移動するのに***OD Previous in context*** 関数または***OD Goto in context*** 関数でROWIDのカラムを用いることはできません。これらのコマンドを使うにはプライマリキーを指定します。

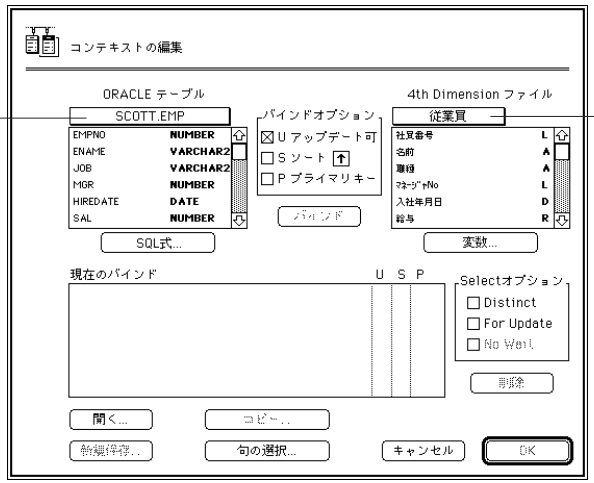
OD Create context dialog

OD Create context dialog **(ログインID;{テーブル名};{4Dテーブル番号}) → 倍長整数**

引数	タイプ	説明
ログインID	倍長整数	接続ID
テーブル名	文字列	デフォルトのOracleのテーブル名
4Dテーブル番号	整数	デフォルトの4th Dimensionテーブルのテーブル番号

OD Create context dialog 関数は、「コンテキストの編集」ダイアログボックスを表示します。このダイアログボックスを使って、4th Dimensionフィールド、変数、または配列をOracleテーブル内のカラムに結合することができます。

「Oracleテーブル」
ポップアップメニュー



「4th Dimensionテーブル」
ポップアップメニュー

コンテキスト定義のための「コンテキストの編集」ダイアログボックスの使用については、第3章の“ ダイアログボックスを使ってコンテキストを作成する ”を参照してください。

「ログインID」は、有効な接続IDでなければなりません。接続IDは、使用を許可されたテーブルに基づいた「Oracle テーブル」ポップアップメニューにどのOracleのテーブルを表示するかを決定するのに使われます。

「テーブル名」は、「Oracle テーブル」ポップアップメニューにデフォルトで表示するOracleのテーブルを指定します。このテーブルは、***OD ADD TO CONTEXT*** コマンドに対して明示的に名前を指定しなかった場合に、このコマンドに対するデフォルトテーブルとして使われます。「テーブル名」は、「ユーザ.テーブル」または「テーブル」の形式でなければなりません。

「4Dテーブル番号」は、「4th Dimension テーブル」ポップアップメニューにデフォルトで表示する4th Dimensionテーブルの番号を指定します。

ユーザがダイアログボックスで「OK」ボタンをクリックすると、***OD Create context dialog*** 関数はコンテキストIDを返します。***OD Activate context*** 関数の呼び出しによってコンテキストをアクティブにする際にこのコンテキストIDを使います。さらに、OKシステム変数に1が設定されます。

ユーザがダイアログボックスで「キャンセル」ボタンをクリックすると、***OD Create context dialog*** 関数は0を返します。そして、OKシステム変数に0が設定されます。

エラーが発生した場合には、***OD Create context dialog*** 関数は-1を返します。

▼ 次の文は、「コンテキストの編集」ダイアログボックスを表示します：

Context_ID = ***OD Create context dialog*** (Login_ID)

参照：OD Create context、OD ADD TO CONTEXT、OD Get options、OD Activate context

OD Create context

OD Create context ({テーブル名}) → 倍長整数

引数	タイプ	説明
テーブル名	文字列	デフォルトのOracleテーブル

OD Create context 関数は、コンテキストの作成を指示してコンテキストのID(識別子)を返します。

この関数の実行は、いずれの4th Dimensionフィールド、変数、または配列もOracleのカラムに結合しません。各バインドに対して***OD ADD TO CONTEXT*** コマンドを呼び出すことによってこれらのバインドを指定します。第8章で説明されている***OD SET OPTIONS*** コマンドを呼び出すことによってコンテキストに対するオプションを設定することができます。

「テーブル名」は、明示的にテーブル名を示さない場合に***OD ADD TO CONTEXT*** に対してデフォルトテーブルとして使われるテーブルを指定します。「テーブル名」は、「ユーザ.テーブル」または「テーブル」の形式でなければなりません。

エラーが発生した場合には、***OD Create context*** 関数は-1を返します。その他の場合には、0より大きい有効なコンテキストの番号を返します。

OD Create context 関数と***OD ADD TO CONTEXT*** コマンドを使ってコンテキストを作成したら、***OD Activate context*** 関数を呼び出してそのコンテキストをアクティブにする必要があります。

▼ 次の例は、“ Scott.EMP ” コンテキストを起動します。

Context_ID = ***OD Create context*** ("Scott.EMP")

また、引数「テーブル名」に複数のテーブルを指定することができます。これを行うには、異なるテーブル名をカンマで区切ります。

▼ 次の例は、部門名を示している最中に “ EMP ” テーブルの従業員に対してコンテキストをアクティブにします：

```
Context_ID:=OD Create context ("EMP","DEPT")
OD ADD TO CONTEXT (Context_ID ; "ENAME" ; >>vEname)
OD ADD TO CONTEXT (Context_ID ; "DNAME" ; >>vDname)
OD SET CLAUSES IN CONTEXT (Context_ID; 2 ; "EMP.DEPTNO = DEPT.DEPTNO)
$rc:=OD Activate context (Login_ID ; Context_ID)
```

参照：OD Create context dialog、OD ADD TO CONTEXT、OD SET OPTIONS、OD Activate context

OD ADD TO CONTEXT

***OD ADD TO CONTEXT* (コンテキストID ; SQL式 ; 4Dオブジェクトポインタ; {オプション})**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID
SQL式	テキスト	Oracleのカラムまたは式
4Dオブジェクトポインタ	ポインタ	4th Dimensionの変数、配列、またはフィールドに対するポインタ
オプション	倍長整数	バインドに対するオプション

OD ADD TO CONTEXT コマンドは、4th Dimensionオブジェクトを Oracleのカラムまたは式にバインドします。そのバインドは、***OD Create context*** 関数によって起動されたコンテキスト定義に追加されます。

「コンテキストID」は、前もって作成された非アクティブなコンテキストIDでなければなりません。

「SQL式」は、Oracleのカラム名（「テーブル.カラム」または「カラム」の形式）か、SQLの式（例えば " SAL*1.15 "）のいずれかを指定します。選択するカラムに対するデフォルトのテーブル名を宣言しなかった場合には、「テーブル.カラム」の形式を使ってカラムが配置されているテーブルを指定する必要があります。***OD Create context dialog*** 関数か、または***OD Create context*** 関数のいずれかを使ってデフォルトのテーブル名を指定することができます。

「4Dオブジェクトポインタ」は、4th Dimensionのフィールド、変数、または配列のポインタです。

「オプション」は、設定したいオプションの合計を表す整数です：

kForUpdate=1	Oracleのカラムのみに適用します。このオプションを使ったカラムは***OD Insert in context*** 関数または***OD Update in context*** 関数によって更新されることが可能です。SQL式が更新されることは不可能です。
kSortAsc=2	Oracle値の昇順ソートを実行します。
kSortDesc=4	Oracle値の降順ソートを実行します。
kKey=8	Oracleのカラムのみに適用します。カラムがコンテキストの行を一意に識別するプライマリキーの一部である、と指定します。プライマリキーの指定は、rowidの特別なカラムが使用可能でない場合に、***OD Previous in context*** 関数または***OD Goto in context*** 関数を使って先頭から終わりまでをアクセスするのとは別の順を追った行アクセスを可能にします。

いずれかのオプションを設定した場合、コンテキストをアクティブにする際に生成されるSELECT文に対する句を定義するのに有効です。

句は、次のように定義されます：

- SELECT：SELECT句は、セレクションで使われるカラムまたは式を含みます。カラムまたは式は、「SQL式」引数で指定することが可能です。複数の「SQL式」引数からの値は、カンマによって区切られています。
- FROM：FROM句は、行が選択されるテーブルを含みます。少なくとも1つのテーブルを指定しなければなりません。「テーブル.カラム」の形式を使って「SQL式」引数でテーブルを指定する場合には、指定するテーブルはFROM句に記入されます。「SQL式」引数でテーブルを指定しない場合には、4D for Oracleは、***OD Create context*** 関数または***OD Create context dialog*** 関数で指定したデフォルトのテーブルを使います。
- FOR UPDATE OF：FOR UPDATE OF句は、***OD SET OPTIONS*** コマンドを使ってこのコンテキストに対してkWithLockオプションが設定されている限り、kForUpdateオプションを持つすべてのカラムの名前を含みます。いずれのカラムもkForUpdateオプションを持たない場合、またはkReadOnlyオプションが***OD SET OPTIONS*** コマンドで設定された場合には、FOR UPDATE OF句は省略されます。
- ORDER BY：ORDER BY句は、kSortASCまたはkSortDescオプションを持つバインドの番号を作成の順に含みます。バインドの番号は、“ ASC ”または“ DESC ”が続きソートが昇順であるか降順であるかを示します。

他の句は、「コンテキストの編集」ダイアログボックスで定義されたコンテキストに応じて表示されます。句は、コンテキストをアクティブにする前に***OD SET CLAUSE IN CONTEXT*** コマンドの呼び出しによって変更することも可能です。

▼次のメソッドは、コンテキスト定義を起動してコンテキスト内でバインドを定義します：

```
kForUpdate := 1                                  ` 句に対する定数を定義する
kSortAsc := 2
kSortDesc := 4
Context_ID = OD Create context ("EMP")           ` コンテキスト定義を起動する
OD ADD TO CONTEXT (Context_ID ; "EMPNO" ; >>[従業員]社員番号)
OD ADD TO CONTEXT (Context_ID ; "ENAME" ; >>[従業員]名前 ; kForUpdate+kSortAsc)
OD ADD TO CONTEXT (Context_ID ; "sal / 1000"; >>vSalaryKF ; kSortDesc)
```

OD Activate context 関数を使ってコンテキストがアクティブになっている場合には、次のSELECT文が Oracleサーバに送られます：

```
SELECT EMPNO, ENAME, SAL/100 FROM EMP ORDER BY 2 ASC, 3 DESC
```

2番目の***OD ADD TO CONTEXT*** コマンドのみがkForUpdateオプションを持つので、ENAMEカラムのみが***OD Update in context*** 関数または***OD Insert in context*** 関数の呼び出しによって更新することができます。

参照：OD Create context、OD Activate context

OD EDIT CLAUSES IN CONTEXT

OD EDIT CLAUSES IN CONTEXT (ログインID ; コンテキストID)

引数	タイプ	説明
ログインID	倍長整数	接続ID
コンテキストID	倍長整数	コンテキストID

OD EDIT CLAUSES IN CONTEXT コマンドは、「SQL句の編集」ダイアログボックスを表示して、指定されたコンテキストの句を編集することができます。

「ログインID」は、アクセス可能なテーブルとカラムのリストをロードするのに使われた有効な接続IDでなければなりません。

「コンテキストID」は、前もって作成された非アクティブなコンテキストIDでなければなりません。

このダイアログボックスを使って、次の句を変更することができます：

- WHERE
- GROUP BY
- HAVING
- CONNECT BY
- START WITH

ユーザがダイアログボックスで「OK」ボタンをクリックすると、OKシステム変数に1が設定されます。「キャンセル」ボタンをクリックした場合には、OKシステム変数に0が設定されます。

▼ 次のメソッドは、コンテキストを作成してコンテキストに対する句の指定を可能にするダイアログボックスを表示します：

```
Context_ID = OD Create context ("EMP")
   ` コンテキストを作成する
OD ADD TO CONTEXT (Context_ID ; "EMPNO" ; >>[従業員]社員番号 ; 0)
OD ADD TO CONTEXT (Context_ID ; "ENAME" ; >>[従業員]名前 ; 0)
   ` バインドを定義する
OD EDIT CLAUSES IN CONTEXT (Login_ID ; Context_ID)
   ` 問い合わせを自分のものにする
```

参照：OD SET CLAUSE IN CONTEXT、OD Get clause in context

OD SET CLAUSE IN CONTEXT

OD SET CLAUSE IN CONTEXT (コンテキストID ; 句番号 ; 句)

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID
句番号	整数	設定する句の番号
句	テキスト	句のテキスト

OD SET CLAUSE IN CONTEXT コマンドは、***OD EDIT CLAUSE IN CONTEXT*** コマンドが画面を使って行なったことをメソッドで実行します。

ユーザがユーザ自身の句を作成できないようにするには、***OD SET CLAUSE IN CONTEXT*** コマンドを使います。ユーザがある程度の句の制御を行なえるようにするけれども、句の入力に対して別のインタフェースを作成したい場合には、「句」引数を含む変数を指定することができます。

OD SET CLAUSE IN CONTEXT コマンドは、コンテキストで句を定義します。「コンテキストID」は前もって作成された非アクティブなコンテキストIDでなければなりません。

「句番号」は、定義したい句を表す番号です：

2 WHERE
3 GROUP BY
4 HAVING
5 CONNECT BY
6 START WITH

「句」は、結合されたキーワードのない句のテキストです（すなわち、“ WHERE ”または他のキーワードがない）。

▼ 次の例は、部門コード（DEPTNO）30の従業員を検索するために使用されるWHERE句を作成します：

OD SET CLAUSE IN CONTEXT (Context_ID ; 2 ; "DEPTNO = 30")

参照：OD EDIT CLAUSES IN CONTEXT、OD Get clause in context

OD Get clause in context

OD Get clause in context (コンテキストID ; 句番号) → テキスト

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID
句番号	整数	取り出す句の番号

OD Get clause in context 関数は、コンテキスト句のテキストを取り出します。

「コンテキストID」は、前もって作成されたアクティブまたは非アクティブなコンテキストIDでなければなりません。

「句番号」は、次の句番号のうちの1つです：

1 FROM
2 WHERE
3 GROUP BY
4 HAVING
5 CONNECT BY
6 START WITH
7 ORDER BY

エラーが発生した場合には、***OD Get clause in context*** 関数は空の文字列を返します。

▼ 次のメソッドは、「コンテキストの編集」ダイアログボックスを使ってユーザがユーザ自身のコンテキストを定義するのを可能にします。このダイアログボックスを使って、WHERE句のような句を追加することができます。WHERE句を取り出した後、ユーザがPresidentに関する情報を取得するのを防ぐために追加の制限がWHERE句に追加されます：

```
Context_ID = OD Create context dialog (Login_ID)              ` コンテキストを作成する
If (Context_ID>0)     ` ユーザが「キャンセル」ボタンをクリックしなかった場合
  clause := OD Get clause in context (Context_ID ; 2)         ` WHERE句を取り出す
  If (clause#"")                                              ` WHERE句が存在しなかった場合
    clause := "(" + clause + ") AND "                         ` ANDを追加する
  End if                                                      ` 連結に備える
  clause := clause + "EMP.JOB <> 'President'"                 ` 条件を追加する
  OD SET CLAUSE IN CONTEXT (Context_ID ; 2 ; clause)    ` WHERE句を定義する
....
End if
```

参照：OD EDIT CLAUSES IN CONTEXT、OD SET CLAUSE IN CONTEXT

OD Save context picture

OD Save context picture (コンテキストID) → ピクチャ

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID

OD Save context picture 関数は、タイプ「ピクチャ」の変数の形式でコンテキストの定義を返します。変数は、取り出し後、「ピクチャ」フィールドに保存されます。

「コンテキストID」は、前もって作成されたアクティブまたは非アクティブなコンテキストIDでなければなりません。

参照：OD Open context picture

OD Open context picture

OD Open context picture (コンテキストピクチャ) → 倍長整数

引数	タイプ	説明
コンテキストピクチャ	ピクチャ	コンテキストの記述を含んだピクチャ変数

OD Open context picture 関数は、「ピクチャ」フィールドまたは変数からコンテキスト定義をロードします。この関数は、新しく作成されたID(識別子)を返します。このコンテキストは、***OD Save context picture*** 関数を使って前もって作成しておく必要があります。

OD Open context picture 関数は、エラーが発生した場合には-1を返します。

OD Activate context 関数を呼び出すことによってコンテキストをアクティブにすることができます。

参照：OD Save context picture、OD Create context dialog、OD Create context

OD SAVE CONTEXT FILE

OD SAVE CONTEXT FILE **(コンテキストID ; ファイル名)**

引数	タイプ	説明
コンテキストID	倍長整数	保存するコンテキストID
ファイル名	テキスト	コンテキスト定義を受け取るファイルの名前

OD SAVE CONTEXT FILE コマンドは、テーブルにあるコンテキストの定義をディスクに保存します。

「コンテキストID」は、前もって作成されてたアクティブまたは非アクティブなコンテキストIDでなければなりません。

「ファイル名」は、作成するドキュメントの名前を指定します。「ファイル名」が空の文字列の場合には、「ファイル保存」ダイアログボックスが表示されます。「ファイル名」が完全なパス名でない場合には、***OD SAVE CONTEXT FILE*** コマンドは、データベースのストラクチャファイルと同じフォルダにファイルを保存します。

同じ名前のドキュメントが既に存在する場合には、上書きされます。

参照：OD Open context file

OD Open context file

OD Open context file **(ファイル名) → 倍長整数**

引数	タイプ	説明
ファイル名	テキスト	コンテキストの記述を含んでいるファイルの名前

OD Open context file 関数は、***OD SAVE CONTEXT FILE*** コマンドを使ってディスク上のファイルに前もって保存されているコンテキスト定義を含んだドキュメントを開きます。

OD Open context file 関数は、新しくオープンされたコンテキストのIDを返します。

「ファイル名」が空の文字列の場合には、「ファイルオープン」ダイアログボックスが表示されます。「ファイル名」が完全なパス名でない場合には、***OD Open context file*** 関数は、データベースのストラクチャファイルを含んでいるフォルダにあるドキュメントを探します。

OD Open context file 関数は、エラーの場合には-1を返して、「ファイルオープン」ダイアログボックスでユーザが「キャンセル」ボタンをクリックした場合には0を返します。

OD Activate context 関数を呼び出すことによってコンテキストをアクティブにすることができます。

参照：OD SAVE CONTEXT FILE、OD Create context dialog、OD Create context

OD Activate context

OD Activate context **(ログインID ; コンテキストID ; {前ロード}) → 整数**

引数	タイプ	説明
ログインID	倍長整数	接続ID
コンテキストID	倍長整数	コンテキストID
前ロード	整数	1=前ロードあり、0=前ロードなし

OD Activate context 関数は、***OD Create context*** 関数、***OD Create context dialog*** 関数によって、あるいはコンテキスト定義を含んでいるピクチャまたはファイルをロードすることによって作成されたコンテキストをアクティブにします。

「ログインID」は、有効な接続IDでなければなりません。「コンテキストID」は、前もって作成された非アクティブなコンテキストIDでなければなりません。

「前ロード」が1の場合、4D for Oracleは***OD Goto in context*** 関数や***OD Previous in context*** 関数、***OD Find in context*** 関数のような直接アクセスできる関数によって使用される内部のキー配列を即座にロードします。

「前ロード」が0または値を指定しなかった場合、キーは必要になった際に（つまり、キーを必要とする関数を呼び出した場合）ロードされます。

ユーザがコンテキストの内部で広範なブラウジングを行ったり、***OD Find in context*** 関数を使用する場合は、引数「前ロード」に1を指定する必要があります。***OD Activate context*** 関数は少し遅くなりますが、その後の処理は速くなります。

OD Activate context 関数は、エラーが発生した場合には-1を返します。その他の場合には1を返します。エラーが発生した場合には、エラー内容を取り出すために***OD Last error*** 関数を使用することができます。

OD Activate context 関数は、カーソルを作成して問い合わせの結果に対応する行を定義します。***OD Activate context*** 関数は、結果行を取り出しません。結果行を表示するには、***OD Next in context*** 関数と***OD Previous in context*** 関数を使わなければなりません。先頭の結果行を取り出すには***OD Next in context*** 関数を使います。

コンテキストがアクティブである限り、***OD ADD TO CONTEXT*** コマンド、***OD EDIT CLAUSES IN CONTEXT*** コマンドまたは***OD SET CLAUSE IN CONTEXT*** コマンドを使って定義を変更することはできません。***OD DEACTIVATE CONTEXT*** コマンドを呼び出して非アクティブにするまでコンテキストはアクティブのままです。

参照：OD DEACTIVATE CONTEXT、OD Create context、OD Last error、OD Next in context、OD Previous in context

OD Find in context

OD Find in context **(コンテキストID) → 倍長整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID

OD Find in context 関数は、プライマリ(主)キーとして使用される4th Dimensionオブジェクトの現在の値と一致するコンテキストの行インデックスを返します。

引数「コンテキストID」はすでにアクティブになっているコンテキストのIDでなければなりません。プライマリキーはコンテキスト設定で指定されている必要があります。

OD Find in context 関数は、現在の値が既存のコンテキスト行と一致してる場合に正の行インデックスを返します。

現在の値がコンテキスト行と一致してない場合は0を返し、エラーが発生した（例えば、指定されたプライマリキーがない）場合は-1を返します。

▼ 次の例は、4th Dimensionのレコード内にOracleの行をロードするためにコンテキストを使用します。**MODIFY SELECTION**コマンドが4th Dimension内のレコードをソート、検索、修正するために呼び出されます。修正のためにユーザがレコードをダブルクリックすると必ず、Oracleサーバ上の対応する行がこのフォームメソッドによって更新されます。

```
Case of
   \ (Before)
     `この4Dレコードと一致するコンテキスト行を検索する
     vIndex:=OD Find in context (Context_ID)
     If (vIndex>0)
        `この4Dレコードが既存のコンテキスト行と一致してるので
        `現在のコンテキスト行に見つけた値をセットする
        $rc:=OD Goto in context (Context_ID ; vIndex)
     End if
   \ (After)
     If (vIndex>0)
        `既存のコンテキスト行と一致した4Dレコードを修正するので
        `対応するOracle行もまた修正される
        $rc:=OD Update in context (Context_ID)
     Else
        `この4DレコードがOracle上にないので、それを作成する
        $rc:=OD Insert in context (Context_ID)
     End if
End case
```

このメソッドは、例えばソート後とかレコードのサブセレクションを作成した後で使用します。これにより、4th Dimension内で複製されるOracleデータを使って処理を行う場合にすばらしい柔軟性を持たせることができます。

▼ 次の例は、特定コンテキストのレコードの中からあるレコードを検索するために***OD Find in context*** 関数を使用します。

```
C_LONGINT (vEmpno)
C_TEXT (vEname)
Context_ID:=OD Create context ("SCOTT.EMP")
OD ADD TO CONTEXT (Context_ID ; "EMPNO" ; >>vEmpno ; kKey)
OD ADD TO CONTEXT (Context_ID ; "ENAME" ; >>vEname)
   `コンテキストをアクティブにし、プライマリキー(EMPNOカラム)の値をロードする
$rc:=OD Activate context (Login_ID ; Context_ID ; 1)
```

```
Repeat
  vEmpno:=Num (Request ("社員番号を入力してください："))
  If (OK=1)
    $index:=OD Find in context (Context_ID)
    If ($index>0)
      $rc:=OD Goto in context (Context_ID ; $index)
      ALERT ("従業員名："+vEname)
    Else
      ALERT ("この番号の従業員はいません。")
    End if
  End if
Until (OK=0)
OD DROP CONTEXT (Context_ID)
```

参照：OD Goto in context、OD ADD TO CONTEXT

OD Next in context

OD Next in context (コンテキストID ; {索引}) → 整数

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID
索引	整数	索引の配列

OD Next in context 関数は、カレント行に続く結果行をロードします。カレント行が定義されていない場合には、***OD Activate context*** 関数の後の場合のように、***OD Next in context*** 関数が先頭の結果行をロードします。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

OD Next in context 関数は、行を使って4th Dimensionオブジェクトを更新します。フィールドを含んでいるバインドに対して、カレントレコードは更新されますが保存はされません。カレントレコードが存在しない場合には、***OD Next in context*** 関数はバインドを無視します。

「索引」は、配列を含むバインドに対してのみ適用されます。「索引」が指定されなかった場合には、***OD Next in context*** 関数は新規カレント行の番号に対応する配列の行を更新します。「索引」が指定された場合には、***OD Next in context*** 関数は、「索引」の配列の行を更新します。「索引」が配列の範囲を超えている場合には、***OD Next in context*** 関数は新規行を配列に追加します。

OD Next in context 関数は、エラーの場合には-1を返します。その他の場合には1を返します。

▼ 次のメソッドは、EMPテーブルと[従業員]テーブルの間にコンテキストを定義して前もって定義された配列の一部である従業員を格納します：

```
Context_ID = OD Create context ("EMP")     ` コンテキストを作成、バインドを定義する
OD ADD TO CONTEXT (Context_ID ;"EMPNO";>>[従業員]社員番号 ; 0)
OD ADD TO CONTEXT (Context_ID ;"ENAME";>>[従業員]名前 ; 0)
OD ADD TO CONTEXT (Context_ID ;"SAL";>>[従業員]給与 ; 0)
err := OD Activate context (Login_ID ; Context_ID)     ` コンテキストをアクティブにする
If (err=1)
  CREATE RECORD ([従業員])                             ` レコードを作成して結果をロード
  err := OD Next in context (Context_ID)               ` 先頭結果行をロードする
  While (err=1)
    If (Find in array (aArray;[従業員]名前)#-1)        ` 従業員が配列にある場合は
      SAVE RECORD ([従業員])                           ` レコードを保存する
      CREATE RECORD ([従業員])                         ` 新規レコードを作成する
    End if
```

```
   err := OD Next in context (Context_ID)          ` コンテキストの次の行に移動する
  End while
  OD DEACTIVATE CONTEXT (Context_ID)               ` コンテキストを非アクティブにする
End if
```

参照：OD Previous in context、OD Activate context

OD Previous in context

OD Previous in context **(コンテキストID ; {索引}) → 整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID
索引	整数	配列の索引

OD Previous in context 関数は、カレント行の前の結果行をロードします。カレント行が定義されていない場合には、***OD Activate context*** 関数の後のように、***OD Previous in context*** 関数は何も行ないません。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

OD Previous in context 関数は、カレント行と結合されている4th Dimensionオブジェクトを更新します。フィールドを含んでいるバインドに対して、カレントレコードは更新されますが保存はされません。カレントレコードが存在しない場合には、***OD Previous in context*** 関数は対応するバインドを無視します。

「索引」は、配列を含んでいるバインドに対してのみ適用されます。「索引」が指定されなかった場合には、***OD Previous in context*** 関数は新規カレント行の番号に対応する配列の行を更新します。

「索引」が指定された場合には、***OD Previous in context*** 関数は、「索引」の配列の行を更新します。「索引」が配列の範囲を超えていた場合には、***OD Previous in context*** 関数は、新規行を追加します。

OD Previous in context 関数は、エラーが発生しなかった場合には1、前の行が存在しなかった場合には0、エラーが発生した場合には-1を返します。

参照：OD Activate context、OD Next in context

OD Goto in context

OD Goto in context **(コンテキストID ; 行番号 ; {索引}) → 整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID
行番号	倍長整数	コンテキストにある行の番号
索引	整数	配列の索引

OD Goto in context 関数は、「行番号」をロードします。「行番号」はカレント行になります。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

「行番号」が行の数より大きいか、あるいは1より小さい場合には、***OD Goto in context*** 関数は何も行なわずに0を返します。

OD Goto in context 関数は、新規カレント行で4th Dimensionオブジェクトを更新します。フィールドを含んでいるバインドに対して、カレントレコードは更新されますが保存はされません。カレントレコードが存在しない場合には、***OD Goto in context*** 関数はバインドを無視します。

「索引」は、配列を含んでいるバインドに対してのみ適用されます。「索引」が指定されなかった場合、***OD Goto in context*** 関数は、新規カレント行がロードされていなければ各配列に行を追加します。ロードされていれば、***OD Goto in context*** 関数は新規カレント行番号を持っている配列の行を更新します。

「索引」が指定された場合には、***OD Goto in context*** 関数は、各配列の「索引」行を更新します。「索引」が配列の範囲を超えていた場合には、***OD Goto in context*** 関数は配列に新規行を追加します。

OD Goto in context 関数は、成功した場合には1、「行番号」が存在しない場合には0、エラーが発生した場合には-1を返します。

参照：OD Number in context、OD Records in context

OD Load rows context

***OD Load rows context* (コンテキストID ; {制限}) → 整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストのID
制限	倍長整数	ロードする最大レコード数

OD Load rows context 関数は、コンテキストによって定義されたすべての行をコンテキストのバインドで指定された4th Dimensionオブジェクトにロードします。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

「制限」は、ロードする行の最大数を指定することを可能にします。

OD Load rows context 関数は、必要なだけの4th Dimensionレコードまたは配列要素を作成します。「制限」が1で、コンテキストがフィールドとのバインドを含んでいる場合には、***OD Load rows context*** 関数はメモリーにレコードを作成して結果行をロードしますが保存はしません（**SAVE RECORD**コマンドは呼び出されません）。

結果が4th Dimensionレコードにロードされる際には、レコードのカレントセレクションが4D for Oracleによって作成されたレコードに設定されます。

OD Load rows context 関数はロードされた行の番号、またはエラーが発生した場合には-1を返します。

参照：OD Activate context、OD Next in context

配列処理

OD Load rows context 関数は、配列処理を使用します。

OD Update in context

OD Update in context **(コンテキストID ; {索引}) → 整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID
索引	整数	配列の索引

OD Update in context 関数は、コンテキストのバインドによって指定された4th Dimensionオブジェクトに基づいたOracleサーバ上のカレント行を更新します。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

「索引」は、配列を含んでいるバインドに対してのみ適用されます。「索引」が指定されなかった場合には、***OD Update in context*** 関数は、カレント行の番号に対応する配列の行を使ってカレント行を更新します。カレント行の番号が配列のサイズより大きい場合には、***OD Update in context*** 関数は-1を返します。

「索引」が指定された場合には、***OD Update in context*** 関数は、「索引」で配列の行を使ってカレント行を更新します。「索引」が配列のサイズより大きい場合には、***OD Update in context*** 関数は-1を返します。

OD Update in context 関数は、" UPDATE ... WHERE " タイプのSQL問い合わせを送信して、コンテキストの一部でしかもkForUpdateオプション（***OD ADD TO CONTEXT*** コマンド参照）が設定されているカラムのみを指定します。

OD Update in context 関数は、成功した場合には1、エラーが発生した場合には-1を返します。

参照：OD Insert in context、OD Delete in context

OD Insert in context

OD Insert in context **(コンテキスID ; {索引}) → 整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID
索引	整数	テーブルの索引

OD Insert in context 関数は、コンテキストのバインドで指定されている4th Dimensionオブジェクトを使ってOracleサーバ上に新規行を作成します。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

「索引」は、配列を含んでいるバインドに対してのみ適用されます。「索引」が指定されなかった場合には、***OD Insert in context*** 関数は、カレント行の番号に対応する配列要素を使って新規行を作成します。行の番号が配列のサイズより大きい場合には、***OD Insert in context*** 関数は-1を返します。

「索引」が指定された場合には、***OD Insert in context*** 関数は、「索引」で配列の行を使って新規行を作成します。「索引」が配列のサイズより大きい場合には、***OD Insert in context*** 関数は-1を返します。

OD Insert in context 関数は、“ INSERT ... ” タイプのSQL問い合わせを送信して、コンテキストの一部でしかもkForUpdateオプション（***OD ADD TO CONTEXT*** コマンド参照）が設定されているカラムのみを指定します。

OD Insert in context 関数は、成功した場合には1、エラーが発生した場合には-1を返します。

OD Insert in context 関数の実行は、カレント行を変更もしないし、コンテキストに新規行を追加もしません。

参照：OD Update in context、OD Delete in context

OD Delete in context

OD Delete in context (コンテキストID) → 整数

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID

OD Delete in context 関数は、Oracleサーバ上のコンテキストのカレント行を削除します。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

カレント行が削除される際に、カレント行が存在しない場合には次の行がカレント行になり、次の行が存在しない場合には前の行がカレント行になります。削除後、それ以上の行がない場合には、カレント行はもはや定義されません。

行の数とカレント行番号は、***OD Delete in context*** 関数を呼び出した後は変更されませんが、Oracleサーバ上の行は物理的に削除されます。***OD Next in context*** 関数、***OD Previous in context*** 関数、または***OD Goto in context*** 関数を使って既に削除された行をロードしようとした場合、4D for Oracleはエラーを返します。

OD Delete in context 関数は、“ DELETE ... WHERE ” タイプのSQL問い合わせを送信します。

OD Delete in context 関数は、成功した場合には1、「行番号」が存在しない場合には0、エラーが発生した場合には-1を返します。

参照：OD Update in context、OD Insert in context

OD Context state

OD Context state **(コンテキストID) → 整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID

OD Context state 関数は、ID(識別子)が指定されているコンテキストの状態を知らせるコードを返します。

次のリストは、リターンコードの意味を説明しています：

0 **コンテキストが正しくありません（作成されたいずれのコンテキストにも対応しません）。**
1 **アクティブでないコンテキストです。**
2 **読み込まれていない行を含むアクティブなコンテキストです。**
3 **すべての行が読み込まれたアクティブなコンテキストです。**

参照：OD Create context、OD Activate context、OD DROP CONTEXT

OD Records in context

OD Records in context **(コンテキストID) → 倍長整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID

OD Records in context 関数は、コンテキストによって定義された行の数を返します。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

OD Records in context 関数によって返された値は、***OD Goto in context*** 関数によって使用可能な最大値です。

OD Records in context 関数は、エラーが発生した場合には-1を返します。

参照：OD Number in context、OD Goto in context

OD Number in context

OD Number in context **(コンテキストID) → 倍長整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID

OD Number in context 関数は、コンテキストにあるカレント行の番号を返します。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

OD Activate context 関数を呼び出すと、カレント行は定義されなくて***OD Number in context*** 関数は0を返します。最初の***OD Next in context*** 関数の後、***OD Number in context*** 関数は1を返します。エラーが発生した場合には-1を返します。

参照：OD Goto in context、OD Records in context

OD DEACTIVATE CONTEXT

OD DEACTIVATE CONTEXT (コンテキストID)

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID

OD DEACTIVATE CONTEXT コマンドは、***OD Activate context*** 関数によって既にアクティブにされたコンテキストを非アクティブにします。

「コンテキストID」は、前もって作成されたアクティブなコンテキストIDでなければなりません。

OD SET CLAUSE IN CONTEXT コマンドを使って、コンテキストを変更することができます。

参照：OD Activate context

OD DROP CONTEXT

OD DROP CONTEXT (コンテキストID)

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID

OD DROP CONTEXT コマンドは、「コンテキストID」によって識別されるコンテキストの定義によって使われたメモリーを解放します。

OD DROP CONTEXT コマンドを呼び出してコンテキストをクローズすると、コンテキストはもはや使用可能ではなくそのIDは無効になります。コンテキストが***OD DROP CONTEXT*** コマンドが呼び出された時点でアクティブである場合には、そのコンテキストが最初に非アクティブになります。

▼ 次のメソッドは、コンテキストの作成を可能にするダイアログボックスを表示してコンテキストをクローズします：

```
Context_ID = OD Create context dialog (Login_ID)
OD DROP CONTEXT (Context_ID)
```

参照：OD Create context、OD Create context dialog、OD DEACTIVATE CONTEXT

OD Reset context

OD Reset context **(コンテキストID) → 倍長整数**

引数	タイプ	説明
コンテキストID	倍長整数	コンテキストID

OD Reset context 関数は、コンテキストの問い合わせを再実行できるのでOracleサーバ上で起こったレコードの削除や最終の追加に注意するよう4D for Oracleに求めます。

引数「コンテキストID」は、すでに作成されてアクティブになっているコンテキストIDでなければなりません。

エラーが発生すると、***OD Reset context*** 関数は-1、それ以外は1を返します。

OD Reset context 関数は、***OD DEACTIVATE CONTEXT*** コマンドと同等の意味を持ちます。しかし、***OD Reset context*** 関数の方が簡単に問い合わせを再実行したり、カーソルや取り出したバッファを保存することができるのでより効果的です。

コンテキストが再度ロードされたキーを使ってアクティブになると、***OD Reset context*** 関数はそのキーを再度ロードします。

注：

- ***OD Reset context*** 関数を実行すると、コンテキストは***OD Activate context*** 関数を呼び出した後と同じ状態にあります。特にカレントの行はもはや定義することはできません。
- コンテキストの再初期化は付随する問い合わせの再実行を意味するので、そのコンテキストを再初期化した後、いくつかのレコードから構成することができます。
- ソートされていないコンテキストを再初期化すると、そのコンテキスト内のレコードはOracleサーバ上での動作に依存した別の順序で表示することができます。
- レコードが***OD Insert in context*** 関数を使用してOracleサーバ上に作成された場合、そのレコードはコンテキストの一部としては見なされません。特に***OD Find in context*** 関数を呼び出した後では、それを見つけることはできません。

▼ 次の例は、この問題を解決するために***OD Reset context*** 関数を使用します。コンテキストは、***OD Next in context*** 関数と***OD Previous in context*** 関数で表示されるダイアログボックスを使用することにより部門コード10に所属する従業員の名前を表示することができます。このコンテキストはダイアログボックスが表示される前に作成され、アクティブになります：

```
Context_ID:=OD Create context ("SCOTT.EMP")
OD ADD TO CONTEXT (Context_ID ; "EMPNO" ; >>vEmpno ; kKey)
OD ADD TO CONTEXT (Context_ID ; "ENAME" ; >>vEname)
OD ADD TO CONTEXT (Context_ID ; "DEPTNO" ; >>vDept)
OD SET CLAUSE IN CONTEXT (Context_ID ; 2 ; "DEPTNO=10")
```

ダイアログボックス内のボタンにより、新しく従業員レコードを作成し、その新しい従業員が部門コード10に所属しているかどうかコンテキスト内で即座に確認することができます。

このボタンのスクリプトは、次のようになります：

```
If (OD Insert in context (Context_ID) = 1)
   `コンテキストが追加されると、部門コード10に所属しているかどうか
   `この新規レコードを考慮するので、そのコンテキストを再初期化する
   $rc:=OD Reset context (Context_ID)
End if
```

これは、***OD Delete in context***関数を使用してレコードを削除したい場合と同じことです。

同じOracleデータベース上で同時に働く人がいて、その従業員が別の部門に異動になったと仮定します。その場合、次のような新規ボタンを使うことにより、ユーザはそのデータベース内での修正を考慮にいれたコンテキストを再構築することができます。

このボタンのスクリプトは、次のとおりです。

```
$rc:=OD Reset context (Context_ID)
```

この章のコマンドは、4th DimensionとOracleサーバ間の通信を管理することを可能にします。ローレベルコマンドを使ってこの通信を管理する際には、Oracleサーバ上のデータを選択、追加、更新、削除するために使うSQL文を指定しなければなりません。

この章のローレベルコマンドを使って、次の事柄を行なうことができます：

- カーソルを作成する（***OD Create cursor***）
- カーソルにSQL文を設定する（***OD Set SQL in cursor***）
- 4th DimensionオブジェクトとOracleオブジェクトの間にバインドを定義する（***OD BIND TOWARDS SQL***、***OD BIND TOWARDS 4D***）
- カーソルを実行する（***OD EXECUTE CURSOR***）
- 結果行をロードする（***OD Load rows cursor***）
- カーソルについての情報を取り出す（***OD Cursor state***、***OD Number rows processed***、***OD Number of columns***、***OD GET COLUMN ATTRIBUTES***、***OD Get column title***）
- カーソルをクローズする（***OD DROP CURSOR***）

OD Create cursor

OD Create cursor **(ログインID) → 倍長整数**

引数	タイプ	説明
ログインID	倍長整数	接続ID

OD Create cursor 関数は、指定された接続に対するカーソルを作成してオープンします。

カーソルはOracleサーバ上のプロセスであるとみなすことができます。作成する各カーソルは、Oracleサーバ上のデータを選択、追加、更新、または削除するSQL文と後で結合されます。同時に複数のカーソルを作成することができますが、あまりに多くのカーソルを作成することは、サーバのレスポンス(応答時間)を遅くする原因になります。

注：接続ごとに可能なカーソルの最大数は、Oracle サーバ上のinit.ora環境設定ファイルのOPEN_CURSORS引数によって設定されます。

「ログインID」は、有効な接続IDでなければなりません。

OD Create cursor 関数は、操作が成功した場合にはカーソルID、エラーが発生した場合には-1を返します。

参照：OD Cursor state、OD DROP CURSOR、OD EXECUTE CURSOR

OD Set SQL in cursor

OD Set SQL in cursor **(カーソルID ; SQLコマンド) → 整数**

引数	タイプ	説明
カーソルID	倍長整数	カーソルID
SQLコマンド	テキスト	実行するSQLコマンド

OD Set SQL in cursor 関数は、SQL文をカーソルに結合するのを可能にします。このコマンドは、SQL文をOracleサーバに送ります。SQL文は、***OD EXECUTE CURSOR*** コマンドを呼び出した際に実行されます。

「カーソルID」は、前もって作成された非アクティブなカーソルIDでなければなりません。

「SQLコマンド」は、カーソルに結合されているSQL文のテキストです。すべてのSQL文は、そのシンタックスが有効である限り受け付けられます。Oracleによってサポートされているの詳細な説明は『SQLランゲージリファレンスマニュアル』を参照してください。

OD Set SQL in cursor 関数は、操作が成功した場合には1、エラーが発生した場合には-1を返します。

SQL文の中に4th Dimensionフィールド、変数、配列の参照を追加することができます。これを行なうには、参照されるオブジェクトの名前を“ﾇ”と“ﾈ”記号によって囲んで追加します。

▼ 次のメソッドは、「[従業員]名前」フィールドとvJobName変数の値を使ってOracleサーバ上に行を作成します：

```
err := OD Set SQL in cursor (Cursor_ID);"INSERT INTO EMP (ENAME, JOB)
VALUES (ﾇ[従業員]名前ﾈ, ﾇvJobNameﾈ)")
OD EXECUTE CURSOR (Cursor_ID)
```

▼ 1つの配列を参照する場合には、***OD EXECUTE CURSOR*** コマンドを呼び出すことによって配列の行数と同じ回数だけ問い合わせを実行します。例えば、次のメソッドは、aNumbersとaNames配列の値を使ってOracleサーバ上に行を作成します：

```
err := OD Set SQL in cursor (Cursor_ID ; "INSERT INTO EMP (EMPNO, ENAME)
VALUES (ﾇaNumbersﾈ, ﾇaNamesﾈ)")
OD EXECUTE CURSOR (Cursor_ID)
```

SQL文で複数の配列を参照している場合には、配列は同じ数の要素を持っていなければなりません。

同じSQL文でフィールド、変数、配列を結合することができます。

しかし、INSERT文またはUPDATE文内で配列や変数、あるいはフィールドの両方を参照している場合には、コマンドは配列の先頭行を使って1回だけ実行します。

OD BIND TOWARDS SQL コマンドを使って後で参照を定義する場合には、“ﾇ”と“ﾈ”記号の間に参照を挿入しないでください。

▼ 例えば、次の文はSQL問い合わせをカーソルに設定します。参照は空白のままなので、バインドの文を実行することによって挿入されるデータ源を後で決定することができます：

```
err := OD Set SQL in cursor (Cursor_ID ; "INSERT INTO EMP (EMPNO, ENAME)
VALUES (ﾇﾈ, ﾇﾈ)")
```

参照：OD EXECUTE CURSOR、OD BIND TOWARDS SQL、OD BIND TOWARDS 4D

OD BIND TOWARDS SQL

OD BIND TOWARDS SQL **(カーソルID ; 参照番号 ; 4Dオブジェクトポインタ)**

引数	タイプ	説明
カーソルID	倍長整数	カーソルID
参照番号	整数	参照番号
4Dオブジェクトポインタ	ポインタ	4Dの変数またはフィールドのポインタ

OD BIND TOWARDS SQL コマンドは、***OD SET SQL in cursor*** 関数を使ってサーバに送信されるSQL文で4th Dimensionオブジェクトの参照を定義することを可能にします。

「カーソルID」は、前もって作成された非アクティブなカーソルIDでなければなりません。

「参照番号」は、参照のカーソルに定義されたSQL文の項目番号です。

「4D オブジェクトポインタ」は、参照するフィールド、変数、または配列のポインタです。

▼ 次のメソッドは、名前が保存されている従業員に対する行をaNumbersとaNames配列から作成します。1番目のメソッドは、***OD Set SQL in cursor*** 文で新規行を作成するのに使われる配列を明示的に決定します：

```
err := OD Set SQL in cursor (Cursor_ID ; "INSERT INTO EMP (EMPNO, ENAME)
VALUES (:aNumbers, :aNames)")
OD EXECUTE CURSOR (Cursor_ID)
```

2番目のメソッドは、一般的な***OD Set SQL in cursor*** 文を実行した後で配列を決定します：

```
err := OD Set SQL in cursor (Cursor_ID; "INSERT INTO EMP VALUES (:, :)")
OD BIND TOWARDS SQL (Cursor_ID ; 1; >>aNumbers)
OD BIND TOWARDS SQL (Cursor_ID ; 2; >>aNames)
OD EXECUTE CURSOR (Cursor_ID)
```

配列ポインタを使うことによって、配列の名前が変更される度にSQL文を更新する必要がなくなります。さらに、4D Insiderのクロスリファレンス(相互参照)機能は明示的に入力された配列名を認識することはできません（その名前は文字列としてのみ認識されるので）が、ポインタによって参照された配列は認識することができます。

参照：OD Set SQL in cursor、OD BIND TOWARDS 4D、OD EXECUTE CURSOR

OD BIND TOWARDS 4D

OD BIND TOWARDS 4D (カーソルID ; カラム番号 ; 4Dオブジェクトポインタ)

引数	タイプ	説明
カーソルID	倍長整数	カーソルID
カラム番号	整数	カラムの番号
4Dオブジェクトポイント	ポインタ	4Dの変数またはフィールドのポインタ

OD BIND TOWARDS 4D コマンドは、***OD Set SQL in cursor*** 関数を使ってSELECT ... タイプのSQLコマンドを送信した後、結果カラムと4th Dimensionオブジェクトの参照間を結合することができます。

「カーソルID」は、前もって作成された非アクティブなカーソルIDでなければなりません。

カラム番号」は、バインドされている結果カラムの項目番号です。

「4Dオブジェクトポインタ」は、カラム番号「カラム番号」の結果を受け取るフィールド、変数、または配列のポインタです。

結果行をロードをするには、***OD EXECUTE CURSOR*** コマンドを呼び出した後、***OD Load rows cursor*** 関数を呼び出す必要があります。

▼ 次のメソッドは、従業員の名前をaNames配列にロードします：

```
err := OD Set SQL in cursor (Cursor_ID ; "SELECT ENAME FROM EMP")
                                        ` 1つの結果カラムを返す
OD BIND TOWARDS 4D (Cursor_ID ; 1 ; >>aNames)
                                        ` 結果カラムを配列に結合する
OD EXECUTE CURSOR (Cursor_ID)           ` 問い合わせを実行する
err := OD Load rows cursor (Cursor_ID)  ` 結果をロードする
```

参照：OD Set SQL in cursor、OD EXECUTE CURSOR、OD BIND TOWARDS SQL、OD Load rows cursor

OD EXECUTE CURSOR

OD EXECUTE CURSOR (カーソルID ; {バッファサイズ})

引数	タイプ	説明
カーソルID	倍長整数	カーソルID
バッファサイズ	倍長整数	バッファサイズとして使用される行の数

OD EXECUTE CURSOR コマンドは、Oracleサーバ上のカーソルに結合されたSQL文を実行します。

「カーソルID」は、前もって作成された非アクティブなカーソルIDでなければなりません。

「バッファサイズ」は、結果を取り出すために使用するバッファサイズを指定することができるようになりました。

実行されているSQL文がSELECT ... タイプの問い合わせ(クエリー)である場合には、***OD EXECUTE CURSOR*** コマンドは、すべての結果行が***OD Load rows cursor*** 関数によってロードされている限り、または***OD CANCEL LOADING*** コマンドが呼び出されていない限り、カーソルをアクティブにします。

また、***OD Load rows cursor*** 関数がSELECTタイプの問い合わせで使用されると、4D for Oracleは一度に複数の行を取り出すために配列処理を利用します。配列処理を使用するには、4D for Oracleは複数の行を保持するためにメモリ内に大きなバッファを割り当てる必要があります。このバッファにより、多くの行を同時に取り出すことができます。このバッファは***OD EXECUTE CURSOR*** コマンドが呼び出された際に割り当てられます。

4D for Oracleは、***OD Load rows cursor*** 関数で取り出される行の数を前もって知ることはできません。バッファサイズの値を指定しないと、4D for Oracleは20行のデータを保持するのに十分な大きさのバッファを割り当てようとします。しかし、これが20以上の行を取得できないことを意味してるわけではありません。4D for Oracleが一度に20行の結果を取り出すことを単に意味してるだけです。つまり、最適なパフォーマンスのためには、このバッファは20より少ない行を取り出すには大きすぎ、20以上の行を取り出すには少なすぎるということです。

引数「バッファサイズ」は、4D for Oracleがユーザの必要なバッファサイズを作るのでデフォルトサイズの20行を無効にすることができます。

バッファは、***OD CANCEL LOADING*** コマンドでカーソルを取り消すと解放されます。また、***OD Load rows cursor*** 関数ですべての結果行を取り出した際にも解放されます。

▼ 次の例は、引数「バッファサイズ」の使用方法を示したものです。

```
$rc:=OD Set SQL in cursor (Cursor_ID ; "SELECT NAME FROM DEPARTMENTS")
   `データベース内に部門コード95があることはわかっているので、
   `バッファを調整するためにこの情報を利用する
OD BIND TOWARDS 4D (Cursor_ID ; 1 ; >>aName)
OD EXECUTE CURSOR (Cursor_ID ; 95)
OD Load rows cursor (Cursor_ID )
$rc:=OD Set SQL in cursor (Cursor_ID ; "SELECT * FROM INVOICE_ITEMS")
   `データベース内に送り状の項目がいくつあるかわからないが、
   `快適なパフォーマンスを得るために大きいサイズをバッファに割り当てる
OD BIND TOWARDS 4D (Cursor_ID ; 1 ; >>aInv_No)
OD BIND TOWARDS 4D (Cursor_ID ; 2 ; >>aInv_Name)
OD EXECUTE CURSOR (Cursor_ID ; 200)
OD Load rows cursor (Cursor_ID )
```

注：使用可能なメモリで指定されたバッファサイズを割り当てることができない場合、4D for Oracleは小さいバッファサイズを使用します。

参照：OD Set SQL in cursor、OD Load rows cursor、OD CANCEL LOADING

OD Load rows cursor

OD Load rows cursor **(カーソルID ; {制限}) → 倍長整数**

引数	タイプ	説明
カーソルID	倍長整数	カーソルID
制限	倍長整数	返す行数

OD Load rows cursor 関数は、***OD EXECUTE CURSOR*** コマンドを使ったSQL問い合わせの実行より結果として生じた行をロードします。

「カーソルID」は、前もって作成されたアクティブなカーソルIDでなければなりません。

「制限」は、取り出す行の最大数を指定します。「制限」が指定されないか0の場合には、***OD Load rows cursor*** 関数はまだロードされていないすべての結果行をロードします。

OD Load rows cursor 関数は、必要なだけのレコードと配列の行を作成します。「制限」が1でフィールドに結果を返している場合には、***OD Load rows cursor*** 関数はメモリーにレコードを作成して結果行を取り出しますが、保存はしません（**SAVE RECORD**コマンドは呼び出されません）。

結果を4th Dimensionレコードにロードしている場合には、テーブルのレコードのカレントセレクションは4D for Oracleによって作成されたレコードに設定されます。

OD Load rows cursor 関数は返された行の数、それ以上の行がない場合には0、エラーが発生した場合には-1を返します。

▼ 次のメソッドは、従業員の名前をvNames変数にロードします：

```
err := OD Set SQL in cursor (Cursor_ID ; "SELECT ENAME FROM EMP")
OD BIND TOWARDS 4D (Cursor_ID ; 1 ; >>vNames)
OD EXECUTE CURSOR (Cursor_ID)
Repeat
n := OD Load rows cursor (Cursor_ID;1)
   If (n>0)
     ALERT (vNames)                          ` 次の名前を表示する
   End if
Until (n<1)
```

参照：OD Set SQL in cursor、OD EXECUTE CURSOR、OD Number rows processed

OD Cursor state

OD Cursor state **(カーソルID) → 整数**

引数	タイプ	説明
カーソルID	倍長整数	カーソルID

OD Cursor state 関数は、ユーザにIDが「カーソルID」であるカーソルの状態を知らせます。

次の表は、リターンコードとその意味の一覧です：

0 **カーソルは無効です（作成されたいずれのカーソルにも一致しません）。**
1 **カーソルは有効ですが、アクティブではありません。**
2 **カーソルは有効でアクティブです（ロードする行がある限り*OD EXECUTE CURSOR* コマンドの後発生します）。**

参照：OD Create cursor、OD DROP CURSOR、OD EXECUTE CURSOR

OD Number rows processed

OD Number rows processed **(カーソルID) → 倍長整数**

引数	タイプ	説明
カーソルID	倍長整数	カーソルID

OD Number rows processed 関数は、SELECTタイプの問い合わせの場合、カーソルの実行後の***OD Load rows cursor*** 関数によってロードされた行数、あるいはINSERT、UPDATE、またはDELETEのタイプの問い合わせの場合、追加、変更、または削除された行数のいずれかを返します。問い合わせの実行が成功した後この関数を使います。

「カーソルID」は、前もって作成されたカーソルIDでなければなりません。

OD Number rows processed 関数は、エラーが発生した場合には-1を返します。

▼ 次のメソッドは、すべてのマネージャにコミッションを与えて、コミッションを受け取っているマネージャの数を返します：

```
err := OD Set SQL in cursor (Cursor_ID ; "UPDATE EMP SET COMM=1000
  WHERE JOB="MANAGER")
OD EXECUTE CURSOR (Cursor_ID)
ALERT (String (OD Number rows processed (Cursor_ID)) + "人のマネージャがコミッションを受け取りました。")
```

参照：OD Set SQL in cursor、OD EXECUTE CURSOR

OD Number of columns

OD Number of columns (カーソルID) → 倍長整数

引数	タイプ	説明
カーソルID	倍長整数	カーソルID

OD Number of columns 関数は、問い合わせにある結果カラムの数を返します。この関数は、***OD Set SQL in cursor*** 関数の実行が成功した後いつでも使うことが可能です。

「カーソルID」は、前もって作成されたカーソルIDでなければなりません。

OD Number of columns 関数は、エラーが発生した場合には-1を返します。

参照：OD Get column title

OD GET COLUMN ATTRIBUTES

OD GET COLUMN ATTRIBUTES (カーソルID ; カラム番号 ; タイプ; サイズ)

引数	タイプ	説明
カーソルID	倍長整数	カーソルID
カラム番号	整数	カラムの番号
タイプ	整数	カラムのタイプ
サイズ	倍長整数	カラムのサイズ

OD GET COLUMN ATTRIBUTES コマンドは、前にカーソルに連結されたSQL問い合わせの結果カラムのタイプとサイズを取り出すのを可能にします。このコマンドは、***OD Set SQL in cursor*** 関数を使ってカーソルに設定されたSQL文の後実行することができます。

「カーソルID」は、前もって作成されたカーソルIDでなければなりません。

「カラム番号」は、属性を知りたいカラムの項目番号を指定します。

「タイプ」は、「カラム番号」カラムのタイプを返します。次の表は各データタイプに対するコードを示しています：

1	CHAR
2	NUMBER
8	LONG
11	ROWID
12	DATE
23	RAW
24	LONG RAW

「サイズ」は、データが***OD Load rows cursor*** 関数によってロードされたかどうかによって「カラム番号」のデータの実際のサイズか、最大サイズかのいずれかが設定されます。データがロードされた場合には、「サイズ」はデータの実際のサイズを返します。データがロードされなかった場合には、「サイズ」はデータの最大サイズを返します。

▼ 次のメソッドは、サーバ上のデータを選択して配列に格納します。配列は、***OD GET COLUMN ATTRIBUTES*** コマンドによって決定されたカラムのタイプに応じて定義されます。行がロードされた後、行の実際のサイズは、***OD GET COLUMN ATTRIBUTES*** コマンドを再度呼び出すことによって決定されます。

```
err := OD Set SQL in cursor (Cursor_ID ; "SELECT * FROM EMP")
                                        `テーブルの先頭カラムを選択する
OD GET COLUMN ATTRIBUTES (Cursor_ID ; 1 ; vType ; vSize)
                                        `カラムのタイプと最大サイズを得る
Case of                                 `オブジェクトを定義して結果を取り出す
   \(vType=1)                           `CHARタイプ
    ARRAY STRING (vSize ; aAlpha ; 0)
    ptr := >>aAlpha
   \(vType=2)                           `NUMBERタイプ
    ARRAY INTEGER (aInteger ; 0)
    ptr := >>aInteger
End case
OD BIND TOWARDS 4D (Cursor_ID ; 1 ; ptr)
OD EXECUTE CURSOR (Cursor_ID)
err := OD Load rows cursor (Cursor_ID ; 1)
OD GET COLUMN ATTRIBUTES (Cursor_ID ; 1 ; vType ; vSize)
                                        `返された行のサイズを取り出す
ALERT ("先頭行、先頭カラムの実際のサイズ：" + String (vSize))
```

参照：OD Set SQL in cursor、OD EXECUTE CURSOR、OD Get column title

OD Get column title

OD Get column title (カーソルID ; カラム番号) → 文字列

引数	タイプ	説明
カーソルID	倍長整数	カーソルID
カラム番号	整数	カラムの番号

OD Get column title 関数は、***OD Set SQL in cursor*** 関数の実行に続いて結果カラムのタイトルを取り出すことができます。

「カーソルID」は、前もって作成されたカーソルIDでなければなりません。「カラム番号」は、知りたいタイトルのカラムの項目番号を指定します。

OD Get column title 関数は、カラムのタイトルまたはエラーの場合には空の文字列を返します。

▼ 次のメソッドは、テーブルのすべてのカラムを選択してカラムタイトルを表示します。

```
err := OD Set SQL in cursor (Cursor_ID ; "SELECT * FROM EMP")
OD EXECUTE CURSOR (Cursor_ID)
max := OD Number of columns (Cursor_ID)          `結果カラムの数を取り出す
For ($i ; 1 ; max)
   ALERT (OD Get column title (Cursor_ID ; $i))
End for
```

参照：OD Set SQL in cursor、OD EXECUTE CURSOR、OD Number of columns

OD DROP CURSOR

OD DROP CURSOR (カーソルID)

引数	タイプ	説明
カーソルID	倍長整数	カーソルID

OD DROP CURSOR コマンドは、***OD Create cursor*** 関数を使って前もって作成されたカーソルにより使用されたメモリーを解放します。

「カーソルID」は、前もって作成されたカーソルIDでなければなりません。

この章のコマンドは、コンテキスト、カーソル、または接続に対する初期設定を行うことができますます。「初期設定」コマンドを使って、次の事柄を指定することができます：

- コンテキストによって生成されたSELECT文にどのキーワードを追加するか（***OD SET OPTIONS***）
- トランザクションをいつロールバックまたはコミットするか（***OD SET OPTIONS***）
- エラーのダイアログをユーザに表示するか（***OD SET OPTIONS***）
- 各データタイプに対してどの値をOracle NULL値と同等に使うか（***OD SET NULL VALUE***）
- 「コンテキストの編集」と「SQL式の編集」ダイアログボックスで使われたOracleのテーブルをメモリに保持するか（***OD LOAD STRUCTURE***）
- Oracleサーバエイリアスの取得にどのファイルを使用するか（***OD SET CONFIGURATION FILE***）

OD SET OPTIONS

OD SET OPTIONS (オブジェクトID ; オプション ; {モード})

引数	タイプ	説明
オブジェクトID	倍長整数	コンテキスト、カーソル、または接続のID
オプション	倍長整数	使いたいオプション
モード	整数	動作

OD SET OPTIONS コマンドは、コンテキスト、カーソル、または接続の動作に対するオプションを指定することができます。「オブジェクトID」は、アクティブでないコンテキストID、アクティブでないカーソルID、接続ID、または“0”のいずれかです。

コンテキストまたはカーソルに対するオプションを設定するには、最初にコンテキストまたはカーソルが非アクティブにされていなければなりません。それ以上取り出す行がない場合にはカーソルは非アクティブであるとみなされます。

「オプション」は、設定したいオプションの合計です。設定可能なオプションは、「オブジェクトID」がコンテキト、カーソル、または接続を参照するかどうかに依存します。

オプション	番号	説明
kAutoRollback	1	**カーソルに対してのみ。SQLエラーの場合に、カレントトランザクションが取り消されなければならないと指定します。**
kAutoCommit	2	**接続に対してのみ。各SQLデータの更新問い合わせ（追加、変更、または削除）の後、カレントトランザクションが登録されなければならないと指定します。**
kNowWait	4	**コンテキストに対してのみ。kWithLockオプションが選択されたコンテキストのオープン時にロックが見つけられた場合にエラーが返されなければならないと指定します。 そして、問い合わせはSELECT ... FOR UPDATE OF ... NOWAITタイプのものです。逆の場合には、デフォルトで、問い合わせはアンロックされるレコードを待ちます。**
kWithLock	8	**コンテキストに対してのみ。コンテキストは結果行をロックしなければならないと指定します。少なくとも1つのバインドがkForUpdateオプションを持っている場合に問い合わせはSELECT ... FOR UPDATE OF ...タイプのものです。**
kDistinctLines	16	**コンテキストに対してのみ。DISTINCTキーワードを（結果行の重複を削除する）コンテキストのSELECT句にすることによってコンテキストがアクティブにされなければならないと指定します。そして、コンテキストは自動的にリードオンリーモードに設定されます。**
kNoDialogErrors	32	**エラーが発生した際に警告を表示しないということをプログラムに知らせます。**
kDeferred	64	**Oracle7に問い合わせを送信している最中にOCIの遅延モードの使用をプログラムに許可します。**
kTextMapping	128	**代用変数を使ってINSERT文またはUPDATE文の中で指定します。4DのC_TEXTタイプはデフォルトのLONGの代わりにVARCHAR(2000)に変換されます。これで、Oracleでサポートされていない1つの問い合わせ当たりにおける複数のLONGカラムの問題を解決します。**

「モード」は、「オプション」によって指定されたオプションを取り替える動作を指示します：

0またはコードなし	**指定されたオプションを設定する**
1	**指定されたオプションを設定解除する**
2	**設定と設定しないの間でオプションを切り替える**

▼ 例えば、次の文はkAutoCommitとkAutoRollbackオプションを設定します：

OD SET OPTIONS (Context_ID ; 4+8)
次の文は、エラー表示モードを切り替えます。そのオプションが前に設定されていた場合には、設定が解除されます。オプションが前に設定されていなかった場合には、設定されます。

OD SET OPTIONS (0 ; 32 ; 2)

遅延モードの利用

OD SET OPTIONS コマンドは、すべてのオブジェクトに対して新しいコード「kDeferred = 64」を利用することができます。このオプションにより、4D for OracleはOracle7のサーバに問い合わせを送信してる最中にOCIの遅延モードを使用することができます。遅延モードは、ステートメントが実際に実行されるまで、遅延処理（SQL文を解析するステップ、入力用の変数をバインドしたり出力用の変数を定義するステップ等）をOCIに通知します。これにより、各ステップは結果として別な方法でネットワークを通してサーバに呼び出されるのでパフォーマンスが向上します。

遅延モードでない場合、4D for Oracleはステートメントを実行する前にステートメントの引数(入出力パラメータのサイズやタイプ)の記述を要求します。4D for Oracleは返されるカラム数がいくつあるとか、タイプチェックやデータタイプの変換を実行するためのデータタイプが何であるかがわかるように任意の記述を要求します。

遅延モードにおいて、SQL文が実行される前にそのステートメント引数の記述が要求されると、せっかくのパフォーマンスの優位性が失われます。それで、最高のパフォーマンスを得るために、4D for Oracleはステートメント引数のタイプとサイズが4th Dimensionのオブジェクトと一致するように見せかけます。

しかし、これは遅延モード内で次のような結果を引き起こします：

- ***OD Execute SQL*** 関数において、4th Dimensionのオブジェクトが結果を受け取るカラムよりも多く指定されると、エラー「ORA-01007」“ Variable not in select list. ” が発生します。
- 遅延モードでは、4D for ORACLEは必要ならばそれ自身の変換を実行するようORACLE7に要求します。そのため、サポートされるデータタイプの変換リストはORACLE7の互換性に依存します。
- 変換はORACLE7によって実行されるので、各国の言語サポートはORACLE7のスキームとクライアント設定に代わる設定（例えば、ORACLEデータを4th Dimensionの文字タイプの変数に挿入する）を使用します。

従って、コーディング時にステートメント引数のタイプやサイズがわからない場合は、遅延モードを使用しないようにしてください。

参照：OD Get options、OD ON ERROR CALL

OD Get options

OD Get options (オブジェクトID) → 倍長整数

引数	タイプ	説明
オブジェクトID	倍長整数	オブジェクトID

OD Get options 関数は、「オブジェクトID」を使ったオブジェクトに対するオプションの設定を返します。「オブジェクトID」はコンテキストID、カーソルID、接続ID、またはデフォルトのオプションとグローバルなオプションを取り出すための0のいずれかです。

OD Get options 関数は、エラーが発生した場合には-1を返します。

▼ 例えば、次のメソッドはkWithLockオプションが設定されているかどうかを調べます。

```
Options := OD Get options (Context_ID)
If ((Trunc(Options / kWithLock) ; 0) %2=1)` kWithLockオプションが選択されている
   …
Else                                    ` kWithLockオプションが選択されていない
   …
End if
```

参照：OD SET OPTIONS

OD SET NULL VALUE

OD SET NULL VALUE (タイプ ; 値)

引数	タイプ	説明
タイプ	整数	変数のタイプ
値	文字	文字に変換されたタイプに対するNULLの同等値

OD SET NULL VALUE コマンドは、4th Dimensionの各データタイプに対してOracleのNULL値を表す定数を指定することを可能にします。定数が4th Dimensionフィールドまたは変数に入力された場合には、対応するOracleの行はNULL値で更新されます。Oracleの行がNULL値を含む場合には、対応する4th Dimensionフィールドまたは変数は定数を表示します。

「タイプ」は、NULLの同等値を設定している4th Dimensionフィールドまたは変数のタイプを指定する整数です。次の表は、フィールドと変数のタイプと数値コードの一覧です。

フィールドまたは変数のタイプ	コード
文字	0
実数	1
テキスト	2
日付	4
ブール	6
整数	8
倍長整数	9
時間	11

「値」は、「タイプ」に対するNULLの同等値です。「値」はどんな文字列定数でもかまいません。

次の表は、それぞれのタイプに対するデフォルトのNULLの同等値を示します：

フィールドまたは変数のタイプ	デフォルトのヌル値
文字	""
実数	"0"
テキスト	""
日付	"00.00.00"
ブール	"0"（すなわち、False）
整数	"0"
倍長整数	"0"
時間	"00:00:00"

▼ 例えば、実数のフィールドまたは変数に対するデフォルトのNULL同等値は“ 0 ”です。次の文を実行することによってNULL同等値を“ -1 ”に変更することができます。

OD SET NULL VALUE (1 ; "-1")

OracleのNULL値が4th Dimensionの実数フィールドまたは変数にロードされる必要がある場合には、その値として-1を受け取ります。-1を含んでいる4th Dimensionの実数フィールドまたは変数に出くわすと、NULL値がOracleに送られます。

4th Dimensionでは、時間と日付は2つの異なったタイプ、「時間」と「日付」のフィールドまたは変数に格納されます。Oracleでは、時間と日付は同じデータタイプDateに結合されています。

次の文を実行します：

OD SET NULL VALUE (4 ; "01.01.01")
OD SET NULL VALUE (11 ; "02:02:02")

実行の後、結果は次のようになります：

SQLから4th Dimensionへ：

SQL	**4th Dimension**
Date = "13-NOV-95 10:56 A.M."	Date := 97.01.13 Time := 10:56:00
Date = NULL	Date := 01.01.01 Time := 02:02:02

4th DimensionからSQLへ：

4th Dimension	SQL
Date := 01.01.01 Time := 02:02:02	Date = NULL
Date := 97.01.13 Time := 10:56:00	Date = "13-NOV-95 10:56 A.M."
Date := 97.01.13 Time := 02:02:02	Date = "13-NOV-95"
Date := 01.01.01 Time := 10:56:00	Date = "10:56 A.M."

OD LOAD STRUCTURE

OD LOAD STRUCTURE (ログインID ; 高速)

引数	タイプ	説明
ログインID	倍長整数	接続ID
高速	整数	0 = テーブルをロードする　1 = テーブルをメモリに保持する

OD SET LOAD STRUCTURE コマンドは、「コンテキストの編集」と「SQL式の編集」ダイアログボックスで使われたOracleのテーブルリストをメモリに保持するのを可能にします。テーブルをメモリに保持することは、ダイアログボックスの表示をより速くします。デフォルトでは、テーブルリストはいずれかのダイアログボックスが表示される度にロードされます。

「ログインID」は、有効な接続IDでなければなりません。

「高速」が0（デフォルト値）の場合には、プログラムは「コンテキストの編集」または「SQL式の編集」ダイアログボックスが表示される度にテーブルをロードします。「高速」が1の場合には、プログラムはリストを一度だけロードして接続が終わるまでメモリに保持します。

4D for Oracleは、各有効な接続に対するテーブルリストをメモリーに格納することができます。

参照：OD Create context dialog、OD EDIT CLAUSES IN CONTEXT

OD SET CONFIGURATION FILE

OD SET CONFIGURATION FILE (ファイル名)

引数	タイプ	説明
ファイル名	文字列	ファイル名

OD SET CONFIGURATION FILE コマンドは、Oracleサーバのエイリアス群を取得するために使用されるファイルを指定します。***OD SET CONFIGURATION FILE*** コマンドを使うことにより、異なるユーザグループに対して複数のエイリアスセットを定義することができます。これらのエイリアスは、***OD Login dialog*** 関数を呼び出すとログインダイアログボックスのポップアップメニューの中に表示されます。

このファイルはテキストタイプで、しかもユーザが使用しているSQL*Netバージョンに対応した“ TNSNAMES.ORA ”ファイルまたは“ config.ora ”ファイルと同じフォルダ内に置かれてなければなりません。このファイルをSimpleTextやノートパッドのようなアプリケーションで編集できますが、“ TNSNAMES.ORA ”または“ config.ora ”ファイルと同じシンタックス規約に準拠している必要があります。この規約に関する詳細は、***OD Login dialog*** 関数の節を参照してください。

もし、***OD SET CONFIGURATION FILE*** コマンドを呼び出さなかったり、または呼び出したとしても引数「ファイル名」に空の文字列を指定した場合は、4D for Oracleはユーザの「Oracle Home」フォルダ内にある“ TNSNAMES.ORA ”または“ config.ora ”ファイルを使用します。

注：SQL*Net 2.0を使用している場合、“ config.ora ”ファイル内で定義されたバージョン1のSQL*Netエイリアスと“ tnsnames.ora ”ファイル内で定義されたバージョン2のSQL*Netエイリアスを一緒にした新しいファイルを指定するために*OD SET CONFIGURATION FILE*コマンドを使用したくなるかもしれません。

この章のコマンドは、4th Dimensionへのデータのロードを取り消したり発生するすべてのエラーを管理することができます。制御コマンドを使って、次の事柄を行なうことができます：

- 行のロードを取り消す（***OD CANCEL LOADING***）
- 最新の4D for Oracleエラーの情報を取り出す（***OD Last error***）
- エラー処理のメソッドをインストールする（***OD ON ERROR CALL***）
- デバッグウインドウをオープンまたはクローズする（***OD OPEN DEBUG WINDOW***、***OD CLOSE DEBUG WINDOW***）
- デバッグウインドウにメッセージを表示する（***OD MESSAGE DEBUG***）
- 接続の情報を取り出す（***OD LOGIN INFORMATION***）

OD CANCEL LOADING

OD CANCEL LOADING (オブジェクトID)

引数	タイプ	説明
オブジェクトID	倍長整数	カーソルまたはコンテキストID

OD CANCEL LOADING コマンドは、アクティブなコンテキストまたは***OD EXECUTE CURSOR*** コマンドが呼び出されたばかりのカーソルに対するデータのロードを中断します。カーソルは非アクティブになり、コンテキストの行数は呼び出しが行なわれる前にロードされた行数に変更されます。

「オブジェクトID」は、アクティブなカーソルIDまたはアクティブなコンテキストIDのいずれかです。

OD OPEN DEBUG WINDOW

OD OPEN DEBUG WINDOW

OD OPEN DEBUG WINDOW コマンドは、4D for Oracleコマンドへの呼び出しのリアルタイムなトレースを提供するウインドウをオープンします。

このウインドウは、次のメニューを含んだメニューバーを持っています：

- **「ファイル」メニュー：**このメニューは印刷、閉じる、ウインドウの内容の保存に対するコマンドを持っています。

■ **「オプション」メニュー：**このメニューは、ウインドウに表示されるデバッグ情報のレベル（メソッドの引数、結果、エラー、メッセージまたは関数の名前だけを表示するかどうか）を設定します 。また、「エラートレース」メニューアイテムでは、エラーを発見するとすぐに4Dの「トレース」モードに切り替えるよう4D for Oracleに要求します。いったん、アクティブになると、このオプションはデバッグウインドウが閉じられた後でもそのまま値を保持します。

参照：OD CLOSE DEBUG WINDOW

OD Last error

OD Last error **({メッセージ} ; {発生源} ; {位置} ; {コマンド} ; {オブジェクトID} ; {プロセス}) → 倍長整数**

引数	タイプ	説明
メッセージ	テキスト	エラーの内容
発生源	倍長整数	エラーの発生源
位置	倍長整数	コマンド内でのエラーの位置
コマンド	テキスト	エラーを起こしたコマンド
オブジェクトID	倍長整数	エラーの原因であるオブジェクトのID
プロセス	倍長整数	エラーの原因であるプロセスの番号

OD Last error 関数は、4D for Oracleの最新のエラー番号を返します。4D for Oracleのエラーコードは「付録 C」で一覧になっています。

「メッセージ」は、エラーの内容を返します。

「発生源」は、エラーの発生源を示すコードを返します。

100 **4th Dimensionからのエラー**
200 **4D for Oracleからのエラー**
300 **Oracleからのエラー**

「位置」は、コマンドテキスト内でのエラーの位置を返します。エラーの位置は、文の始まりから数えたエラーの先頭文字の番号です。

「コマンド」は、SQLコマンドによってエラーが発生した場合に、エラーの原因となったSQLコマンドのテキストを返します。

「オブジェクトID」は、エラーの原因となった接続、コンテキスト、またはカーソルのIDを返します。

「プロセス」は、エラーの原因となった4th Dimensionプロセスの番号を返します。

注：接続IDは[10000-99999]、カーソルIDは[1000-9999]、コンテキストIDは[100-999]の間にあります。

参照：OD ON ERROR CALL

OD ON ERROR CALL

OD ON ERROR CALL (メソッド名)

引数	タイプ	説明
メソッド名	文字列	エラー処理メソッドの名前

OD ON ERROR CALL コマンドは、エラーが発生する度に実行されるエラー処理メソッドをインストールします。これにより可能な実行エラーを制御してデフォルトのエラー処理を上書きすることができます。

「メソッド名」は、インストールするメソッドの名前です。

デフォルトの結果を返すには、***OD ON ERROR CALL*** ("")のように空の文字列を渡します。

4D for Oracleはメソッドに6つの引数を渡します：

引数	タイプ	説明
$1	倍長整数	エラーの原因であるオブジェクトのID
$2	倍長整数	エラーコールを開始したエラーのコード
$3	テキスト	エラーの説明
$4	倍長整数	エラーの発生源（100;200;300）
$5	倍長整数	コマンド内でのエラーの位置、コマンドテキストの始まりから数えたエラーの先頭文字の番号
$6	テキスト	エラーの原因であるコマンドの名前

データベースをコンパイルする場合には、**C_LONGINT**コマンドと**C_TEXT**コマンドを使って変数$1から$6を宣言しなければなりません。

デフォルトで、どのようなエラー処理メソッドもインストールされていなくてkNoErrDialogオプション（***OD SET OPTIONS***コマンド参照）も設定されていない場合には、すべてのエラーは4D for Oracleによって受け取られてエラーを示す警告が表示されます。

参照：OD Last error

OD CLOSE DEBUG WINDOW

OD CLOSE DEBUG WINDOW

OD CLOSE DEBUG WINDOW コマンドは、***OD OPEN DEBUG WINDOW*** コマンドを使って既にオープンされたデバッグウインドウをクローズします。

参照：OD OPEN DEBUG WINDOW

OD MESSAGE DEBUG

OD MESSAGE DEBUG (テキスト)

引数	タイプ	説明
テキスト	文字列	メッセージのテキスト

OD MESSAGE DEBUG コマンドは、***OD OPEN DEBUG WINDOW*** コマンドによって既にオープンされたデバッグウインドウに「テキスト」を表示します。

OD LOGIN INFORMATION

OD LOGIN INFORMATION ({ログインID})

引数	タイプ	説明
ログインID	倍長整数	接続ID

OD LOGIN INFORMATION コマンドは、接続の状態と、***OD OPEN DEBUG WINDOW*** コマンドによって既にオープンされたデバッグウインドウにすべてのアクティブなカーソルとコンテキストを表示します。

「ログインID」が指定されない場合には、すべてのアクティブな接続に対してリストが作成されます。

4D for Oracleは、4th Dimensionのフィールド、変数、配列とOracleのカラムと式をバインドすることを可能にします。4th DimensionとOracleのデータタイプは同じではないので、4D for Oracleは柔軟性を最大限に発揮し、あるタイプから別のタイプへデータを変換します。

例えば、次の問い合わせでは、「Number」タイプであるEMPNOカラムをタイプが「文字」、「テキスト」、「整数」、「倍長整数」、「実数」、「ブール」、または「時間」である4th Dimensionのフィールド、変数、あるいは配列にバインドすることができます：

```
SELECT EMPNO FROM EMP
```

この変換は、逆方向でも同じように行なうことが可能です。

次の表は、変換処理を一覧にしたものです。最初の表は、Oracleから4th Dimensionへの変換の一覧です。2番目の表は、逆方向、4th DimensionからOracleへの変換の一覧です。

Oracleから4th Dimensionへ

Oracle	4th Dimension	変換の特徴
CHAR	文字	CHARが255バイトであるのに対して文字は80バイト以上であることはできません。文字列は切り捨てられます。
	テキスト	制限はありません。
	整数	4D for Oracleは整数をある程度保持します。結果が[-32768;32767]以外の場合には、エラーが発生します。
	倍長整数	4D for Oracleは整数をある程度保持します。結果が[-2147483648;2147483647]以外の場合には、エラーが発生します。
	実数	SANEの変換に限られます。
	ブール	文字列="1"の場合にはTrue、それ以外はFalseです。
	日付	常に00.00.00へ変換します。
	時間	常に00:00:00へ変換します。
	ピクチャ	常に空ピクチャへ変換します。

Oracle	4th Dimension	変換の特徴
NUMBER	文字	制限はありません。
	テキスト	制限はありません。
	整数	4D for Oracleは整数をある程度保持します。結果が[-32768;32767]以外の場合には、エラーが発生します。
	倍長整数	4D for Oracleは整数をある程度保持します。結果が[-2147483648;2147483647]以外の場合には、エラーが発生します。
	実数	SANEの変換に限られます。
	ブール	0である場合にはFalse、それ以外はTrueです。
	日付	常に00.00.00へ変換します。
	時間	午前0時以降の秒の数値として扱われます。
	ピクチャ	4D for Oracleは整数をある程度保持します。結果が[-214748368;2147483647]以外の場合には、エラーが発生します。 常に空のピクチャへ変換します。
DATE	文字	4Dの省略型の日付フォーマット + スペース + 4Dの省略型の時間フォーマット
	テキスト	4Dの省略型の日付フォーマット + スペース + 4Dの省略型の時間フォーマット
	整数	常に0へ変換します。
	倍長整数	常に0へ変換します。
	実数	常に0へ変換します。
	ブール	常にFalseです。
	日付	Oracleの日付は読み込まれません。
	時間	Oracleの時間は読み込まれません。
	ピクチャ	常に空のピクチャへ変換します。
LONG	文字	文字列変数は255バイト、文字フィールドは80バイトで切り捨てられます。
	テキスト	制限はありません。
	整数	4D for Oracleは整数をある程度保持します。結果が[-32768;32767]の範囲外である場合には、エラーが発生します。
	倍長整数	4D for Oracleは整数をある程度保持します。結果が[-2147483648;2147483647]の範囲外である場合には、エラーが発生します。
	実数	SANEの変換に限られます。
	ブール	文字列= “ 1 ” の場合にはTrueで、それ以外の場合にはFalseです。
	日付	常に00.00.00へ変換します。
	時間	常に00:00:00へ変換します。
	ピクチャ	常に空ピクチャへ変換します。

Oracle	4th Dimension	変換の特徴
RAW	文字	CHARとして扱われます。ROWが256バイトであるのに対して文字は80バイト以上であることはできません。文字列は切り捨てられます。
	テキスト	LONGとして扱われます。制限はありません。
	整数	常に0へ変換します。
	倍長整数	常に0へ変換します。
	実数	常に0へ変換します。
	ブール	常にFalseへ変換します。
	日付	常に00.000.0へ変換します。
	時間	常に00:00:00へ変換します。
	ピクチャ	常に空ピクチャへ変換します。
LONG RAW	文字	文字列変数は255バイト、文字フィールドは80バイトで切り取られます。
	テキスト	LONGとして扱われます。制限はありません。
	整数	常に0へ変換します。
	倍長整数	常に0へ変換します。
	実数	常に0へ変換します。
	ブール	常にFalseへ変換します。
	日付	常に00.00.00へ変換します。
	時間	常に00:00:00へ変換します。
	ピクチャ	ピクチャは4th Dimensionのピクチャフォーマットでなければなりません。
ROWID	文字	bbbbbbbb.llll.ffff bbbbbbbb -> ファイルでのブロック番号 llll -> 行番号 ffff -> ファイル番号
	テキスト	bbbbbbbb.llll.ffff bbbbbbbb -> ファイルでのブロック番号 llll -> 行番号 ffff -> ファイル番号
	整数	常に0へ変換します。
	倍長整数	常に0へ変換します。
	実数	常に0へ変換します。
	ブール	常にFalseへ変換します。
	日付	常に00.00.00へ変換します。
	時間	常に00:00:00へ変換します。
	ピクチャ	常に空のピクチャへ変換します。

4th DimensionからOracleへ

4th Dimension	Oracle	変換の特徴
文字	すべてのタイプ	Oracleは文字列をそのまま受け取ります。制限はSQLと同様です。
テキスト	すべてのタイプ	Oracleは文字列をそのまま受け取ります。制限はSQLと同様です。
実数	すべてのタイプ	文字列の通常の4Dフォーマットへの変換の後、Oracleは文字列をそのまま受け取ります。制限はSQLと同様です。
整数	すべてのテイプ	文字列への変換後、Oracleは文字列をそのまま受け取ります。制限はSQLと同様です。
倍長整数	すべてのタイプ	文字列への変換後、Oracleは文字列をそのまま受け取ります。制限はSQLと同様です。
日付	Date	Oracleの日付が現在の年の1月1日に設定されます。
時間	Date	Oracleの時間が12:00:00に設定されます。
ブール	すべてのタイプ	文字列への変換後（Falseに対しては“0”、Trueに対しては“1”）、Oracleは文字列をそのまま受け取ります。制限はSQLと同様です。
ピクチャ	LONG RAW	ピクチャを4Dの内部フォーマットで保存します。LONG RAW以外のタイプはサポートされていません。

4D for Oracleの関数によって返される値　B

次の表は、4D for Oracleの関数が返す値を一覧にしたものです。

関数名	成功	エラー	取り消しまたは行が不明
ログイン			
OD Login dialog	**ログインID**	-1	0
OD Login	**ログインID**	-1	
OD Login state	**状態**		
OD Execute SQL	**カラム数**	-1	
OD Clone file	1	-1	0
コンテキスト			
OD Create context dialog	**コンテキストID**	-1	0
OD Create context	**コンテキストID**	-1	
OD Get clause in context	**句のテキスト**	**空の文字列**	
OD Save context picture	**ピクチャ**	**空のピクチャ**	
OD Open context picture	**コンテキストID**	-1	
OD Open context file	**コンテキストID**	-1	0
OD Activate context	1	-1	
OD Context state	**状態**		
OD Next in context	1	-1	0
OD Previous in context	1	-1	0
OD Number in context	**行番号**	-1	0
OD Goto in context	1	-1	0
OD Records in context	**行数**	-1	
OD Update in context	1	-1	
OD Insert in context	1	-1	
OD Delete in context	1	-1	
OD Load rows context	**行数**	-1	0

関数名	成功	エラー	取り消しまたは行が不明
ローレベル			
OD Create cursor	**カーソルID**	-1	
OD Cursor state	**状態**		
OD Set SQL in cursor	1	-1	
OD Load rows cursor	**行数**	-1	0
OD Number rows processed	**行数**	-1	
OD Get column title	**カラムタイトル**	**空の文字列**	
OD Number of columns	**カラム数**	-1	
環境設定			
OD Get options	**オプション**	-1	
制御			
OD Last error	**最新のエラー**		

次の表は、4D for Oracleで発生するエラーの一覧です。

エラー番号	説明
1001	ログインが正しくありません。
1002	カーソルが正しくありません。
1003	カーソルが解放されていません。
1004	コンテキストが正しくありません。
1005	コンテキストがアクティブではありません；最初にアクティブにしなければなりません。
1006	コンテキストは既にアクティブになっています。
1007	最初にコンテキストを非アクティブにしなければなりません。
1008	コンテキストは既に非アクティブになっています。
1009	このコンテキストを使ってデータを更新することはできません。
1010	このコンテキストでは何もバインドされていません。
1011	サポートされていないデータタイプです。
1012	“ Oracle Home/NETWORK/ADMIN ” で “ TNSNAMES.ORA ” ファイルを見つけることができません。システム定義を調べてください。
1013	未使用
1014	未使用
1015	“ TNSNAMES.ORA ” ファイルを読み込むことができません。システム定義とイントールを調べてください。
1016	“ TNSNAMES.ORA ” ファイルをクローズすることができません。
1017	このレコードは削除されています。
1018	これ以上メモリを割り当てることができません。
1019	内部エラーです。もう一度試してください。
1020	このエラーのテキストを獲得することはできません。
1021	配列が宣言されていないか、あるいは二重引用符がありません。
1022	このコマンドに対する問い合わせが必要です。
1023	無効なカラム番号またはヌルのカラム番号です。
1024	引数として渡された参照は正しくないか、あるいは無効です。
1025	各カラムは既にバインドされています。
1026	このバインドに対する問い合わせがありません。
1027	問い合わせの実行後のバインドはできません。
1028	このカーソルに対するバインドがありません。

エラー番号	説明
1029	Oracleのテーブルは複数のLONGカラムを持つことはできません（1つのテキストまたは1つのピクチャ）。
1030	このコンテキスト定義のファイルまたはピクチャを読み込むことができません。
1031	このコンテキストは無効であるか、あるいは変更されています。
1032	この番号に対する句がありません。
1033	Oracleは4つより多い接続はサポートしません。
1034	フィールドのないファイルをクローンすることはできません。
1035	サブファイルのクローンはサポートされていません。
1036	エラー -192、リソースをロードすることができません。
1037	システムフォルダを見つけることができません。
1038	エラーメッセージを見つけることができません。システム定義を調べてください。
1039	4Dタイプidが範囲外です。
1040	“ﾇ“ と “ﾈ” は2つ1組で使用されなければなりません。
1041	参照されるファイルまたはフィールドは存在しません。
1042	バインドではポインタのポインタまたはポインタの配列はサポートされていません。
1043	WHERE句はピクチャ上にバインドを含むことはでいません。
1044	このバインドはサポートされていないデータタイプの変換を必要とします。
1045	このファイルを読み込むことはできません。破損している可能性があります。
1046	内部エラーです。ACI社テクニカルサポートに連絡してください。
1047	ポインタ以外のものとバインドが行なわれました。バインドは無効です。
1048	この名前のOracleのテーブルは既に存在します。
1049	ログインパラメータが正しくありません。ログインダイアログを使ってもう一度入力してください。
1050	バインドのストラクチャを作成することができません。おそらくメモリ不足のせいです。
1051	メモリ不足のためデバッグウインドウをオープンすることはできません。
1052	SQL文の “ FROM ” 句を定義することはできません。おそらくコンテキストがSQL式だけのバインドを含んでいるせいです。
1053	メモリ不足のため1行を取り出したり、送ったりすることができません。
1054	このバージョンの4D for Oracleは4th Dimension バージョン3.0.2、4D Server バージョン1.0.2またはそれ以前のバージョンのみと一緒に実行することが可能です。
1055	オブジェクトが正しくありません。使用中であるか、あるいはこのオブジェクトIDに対して無効なオプションです。
1056	*OD Load rows cursor* 関数の前に*OD EXECUTE CURSOR* コマンドが送信されなければなりません。
1057	複数テーブル上のコンテキストはブラウズまたは変更を実行することはできません。
1058	4D for Oracle Preferencesファイルをオープンすることはできません。このファイルは既にオープンされているか、あるいは作成することができませんでした。プレファレンスは保存されません。

エラー番号	説明
1059	エラーハンドラに対するプロシージャ名が正しくありません。
1060	このコマンドに対するカレント行はありません。コンテキストをアクティブにした後*OD Next in context*関数を呼び出して1番目の行を取り込んでください。
1061	複数テーブルのコンテキストを変更することはできません。
1062	このコンテキストには更新可能なバインドはありません；更新や追加は無効となります。
1063	ファイルが見つかりません。
1064	コンパイルされたデータベースでは「コンテキスト編集」ダイアログボックスを使って作成されなかったコンテキスを保存することはできません。
1065	4th Dimension内のすべてのレコードを作成することはできません。
1066	4th Dimension内のすべてのレコードを保存することはできません。
1067	テーブル以外の別のオブジェクトが同じ名前を持っています。
1068	この変数はピクチャではないか、あるいは空のピクチャです。
1069	このピクチャはコンテキスト定義を含んでいません。
1070	未使用
1071	未使用
1072	未使用
1073	このピクチャは大きすぎるか（64 Kbまで）、あるいは無効です。
1074	これはピクチャではありません。
1075	指定された行は存在しません。
1076	制限が範囲外です。
1077	これは、デモバージョンです。
1078	SQL式が長すぎます（最大255バイトまで）。
1079	4D変数がありません。
1080	ピクチャはプライマリキーとして使用できません。
1081	このコンテキスト用に定義されたプライマリキーがありません。
1082	Oracle Homeの“ Drivers ”フォルダ内に“ oci7 ”ドライバがありません。4D for Oracleは、このファイルなしでは機能しません。
1083	互換モードが正しくありません。
1084	記述用のオブジェクト名が正しくありません。
1085	このオブジェクトの記述を取得するにはメモリが足りません。
1086	配列引数のタイプが正しくありません。
1087	このOracleドライバをロードするにはメモリが足りません。 4D for Oracleはこれらのドライバなしに操作することはできません。
1088	予期しないログインIDです。
1089	4DORACLE.INIのシステム定義またはOracleクライアントのインストールを検査するORACLE.DLLをロードすることができません。 4D for Oracleはこれらのドライバなしに操作することはできません。
1090	ACI / 4DORACLE.INIファイルの中でORACLE.DLLの名前を見るけることができません。

4DORACLE.INIファイル

シングルユーザ環境

“ 4DORACLE.INI ” ファイルは、4D for Oracleを機能させるために必要不可欠なファイルです。このファイルをシステムフォルダ（例えば、Windowsフォルダ）内の「ACI」フォルダの中にコピーする必要があります。もし、ACIフォルダ内に存在しない場合は、“ 4DORACLE.INI ” ファイルを作成する必要があります。このフォルダは、ユーザ自身の手によって作成したり、または4Dを起動することによって自動的に作成することもできます。

クライアント／サーバ環境

クライアント／サーバ環境では、“ 4DORACLE.INI ” ファイルはクライアントワークステーション上の「ACI」フォルダ内に配置されている必要があります。また、Oracleドライバが4D for Oracleを使用する各クライアントワークステーション上にインストールされている点に注意してください。

4DORACLE.INIファイルの内容

“ 4DORACLE.INI ” ファイルは、Windows上の他の “ *.INI ” ファイルと同じ内容を持っています。つまり、“ 4DORACLE.INI ” ファイルはインストールされたOracleドライバのDLL (Dynamic Kink Kibrary)の名前が指定された1つのセクションだけで構成されています。この名前はOracleがインストールされたフォルダ（通常は「ORACLE \ BIN」フォルダ）の中で見つけることができます。

下記の “ 4DORACLE.INI ” ファイルの例では、“ ORANT71.DLL ” がWindows NT用にインストールされたOracle DLLの名前です。

```
[INIT]
DLLNAME=ORANT71.DLL
```

Windows3.1の特殊事項

シングルユーザ環境

Windows3.1は16ビット環境なので、インストールされるOracleドライバも16ビット用のものです。4D for Oracleがこの16ビット用のドライバと交信（やり取り）でき、しかも十分に機能できるようにするには、「ACI」フォルダの中に “ OCI_W16.DLL ” ファイルをコピーする必要があります。

クライアント／サーバ環境

クライアント／サーバ環境では、4D Clientは4D Serverではなく4D for Oracleのコードを実行するので、“ OCI_W16.DLL ” ファイルがクライアントワークステーション上の「ACI」フォルダ内に配置されている必要があります。

記号

数字

A

C

D

F

G

H

I

K

M

N

O

キ、き

ク、く

ケ、け

コ、こ

サ、さ

シ、し

ス、す

セ、せ

ソ、そ

タ、た

チ、ち

テ、て

ト、と

ハ、は

ヒ、ひ

フ、ふ

へ、へ

ホ、ほ

マ、ま

メ、め

ル、る

レ、れ

ロ、ろ

コマンド索引