

Pervasive PSQL v11 SP1

Tutorials and Guide to Samples

PERVASIVE®

免責事項

Pervasive Software Inc. は、本ソフトウェアおよびドキュメントの使用を、利用者またはその会社に対して「現状のまま」で、かつ同梱の使用許諾契約書に記載の契約条件によってのみ許諾するものです。Pervasive Software Inc. は、いかなる場合にも本ソフトウェアおよび本マニュアルに記載された内容に関するその他の一切の保証を、明示的にも黙示的にも行いません。Pervasive Software Inc. は、市場性、権利、特定の目的に対する適合性、あるいは一連の取引業務や職業的な使用に関する問題などに対し、一切の保証を行わないことを明示するとともに、利用者およびその会社がこれに同意したものとします。

商標

Btrieve、Client/Server in a Box、Pervasive、Pervasive Software および Pervasive Software のロゴは、Pervasive Software Inc. の登録商標です。

Built on Pervasive Software、DataExchange、MicroKernel Database Engine、MicroKernel Database Architecture、Pervasive.SQL、Pervasive PSQL、Solution Network、Ultralight、ZDBA は Pervasive Software Inc. の商標です。

Microsoft、MS-DOS、Windows、Windows 95、Windows 98、Windows NT、Windows Me、Windows 2000、Windows Server 2003、Windows 2008、Windows 7、Windows XP、Win32、Win32s、および Visual Basic は、Microsoft Corporation の登録商標です。

NetWare および Novell は Novell, Inc. の登録商標です。

NetWare Loadable Module、NLM、Novell DOS、Transaction Tracking System、TTS は、Novell, Inc. の商標です。

Sun、Sun Microsystems、Java、および Sun、Solaris、Java を含むすべての商標やロゴは、Sun Microsystems の商標または登録商標です。

すべての会社名および製品名は各社の商標または登録商標です。

© Copyright 2011 Pervasive Software Inc. All rights reserved. このマニュアルの全文、一部に関わりなく複製、複写、配布をすることは、前もって発行者の書面による同意がない限り禁止します。

本製品には、Powerdog Industries により開発されたソフトウェアが含まれています。

© Copyright 1994 Powerdog Industries. All rights reserved.

本製品には、KeyWorks Software により開発されたソフトウェアが含まれています。

© Copyright 2002 KeyWorks Software. All rights reserved.

本製品には、DUNDAS SOFTWARE により開発されたソフトウェアが含まれています。

© Copyright 1997-2000 DUNDAS SOFTWARE LTD. All rights reserved.

本製品には、Apache Software Foundation Foundation (<http://www.apache.org/>) により開発されたソフトウェアが含まれています。

本製品ではフリー ソフトウェアの unixODBC Driver Manager を使用しています。これは Peter Harvey (pharvey@codebydesign.com) によって作成され、Nick Gorham (nick@easysoft.com) により変更および拡張されたものに Pervasive Software が一部修正を加えたものです。Pervasive Software は、unixODBC Driver Manager プロジェクトの LGPL 使用許諾契約書に従って、このプロジェクトの現在の保守管理者にそのコード変更を提供します。unixODBC Driver Manager の Web ページは www.unixodbc.org にあります。このプロジェクトに関する詳細については、現在の保守管理者である Nick Gorham (nick@easysoft.com) にお問い合わせください。

GNU Lesser General Public License (LGPL) は本製品の配布メディアに含まれています。LGPL は www.fsf.org/licenses/licenses/lgpl.html でも見ることができます。

Tutorials and Guide to Samples

一般リリース 2011 年 7 月

目次

このマニュアルについて	vii
情報の参照先	viii
このマニュアルの読者	ix
このマニュアルの構成	x
表記上の規則	xi
1 Pervasive SDK コード サンプルの概要	1
SDK コード サンプルの概要	2
Pervasive PSQL SDK に含まれるサンプル	3
Video Store アプリケーション	3
Pervasive エンジンへのアクセス	4
2 Microsoft Access での Pervasive データの使用	5
レッスン 1：環境を設定する	6
必要なソフトウェアをインストールする	6
サンプル データベースをコピーする	6
新規データベース名および DSN を作成する	7
データ辞書ファイルを作成する	8
チュートリアル用のデータベースを作成する	9
レッスン 2：テーブルにリンクする	10
レッスン 3：テーブルをインポートする	13
レッスン 4：SQL パススルー クエリを作成する	16
まとめ	20
3 Microsoft Visual Basic での Pervasive データの使用	21
レッスン 1：環境を設定する	22
必要なソフトウェアをインストールする	22
サンプル データベースをコピーする	22
新規データベース名および DSN を作成する	23
データ辞書ファイルを作成する	23
チュートリアル用のプロジェクトを作成する	24
レッスン 2：ADO データ コントロール (ADODC) を使用する	26
レッスン 3：ActiveX Data Object でコードを記述する	33
まとめ	37

4	Pervasive PSQL を Delphi で使用	39
	レッスン 1 : Delphi で ActiveX を使用する	40
	Delphi IDE に ActiveX インターフェイスをインストールする	40
	Delphi で VAccess コントロールを使用する	41
	VAList ボックス コントロールを使用する	44
	VAText コントロールを使用する	46
	2つの VAccess コントロールを結合する	47
	レッスン 2 : Delphi で Btrieve API を使用する	49
	基本的なファイル操作を実行する	49
	データの変更	53
	データの取得	54
	レッスン 3 : Delphi で ODBC を使用する	57
5	Pervasive PSQL を Visual Basic で使用	63
	レッスン 1 : Visual Basic で ActiveX を使用する	64
	レッスン 2 : Visual Basic で Btrieve API を使用する	72
	基本的なファイル操作を実行する	74
	データの変更	78
	レッスン 3 : Visual Basic で ODBC を使用する	79
6	ActiveX を使用する Pervasive PSQL アプリケーションの作成	87
	ActiveX を使用した Visual Basic の例	88
	レッスン 1 : データ ブラウザーを作成する	89
	VB プロジェクトへ Pervasive ActiveX インターフェイスへ追加する	89
	VAccess コントロール フォームを作成する	90
	Pervasive ActiveX インターフェイスのプロパティを設定する	91
	バウンド コントロールのあるフォームを作成する	94
	アプリケーションのテスト	100
	レッスン 2 : 更新用フォームを作成する	101
	フォームの追加と表示	101
	アプリケーションのテスト	104
	レッスン 3 : 2つのテーブルからのレコードを結合する	105
	VAccess コントロールの追加	105
	VAccess コントロールの結合	106
	個人データ フォームの追加と表示	107
	アプリケーションのテスト	109
7	Btrieve API の言語インターフェイス	111
	C/C++	112

C++ プログラムの例	112
Btrieve Login の C++ プログラムの例	123
サンプルプログラムのコンパイル、リンク、実行	130
C++ Builder アプリケーションのコンパイル	134
COBOL	147
Delphi	150
プログラムの例	150
サンプルプログラムのコンパイル、リンク、実行	162
Pascal	163
プログラムの例	163
Pascal インターフェイスを使用したコンパイルとリンク	174
Visual Basic	176
プログラムの例	176
サンプルプログラムのコンパイル、リンク、実行	193

このマニュアルについて

このマニュアルでは、Microsoft Access、Visual Basic、ActiveX、および Borland Delphi を使用して Pervasive データを表示および操作する方法について説明します。また、ODBC（Open Database Connectivity）および SQL クエリを使用して Pervasive データにアクセスする方法についても説明します。このマニュアルでは、作業手順やチュートリアル、さらに、新しい概念に対する理解を深めるためのサンプルプログラムが用意されています。

情報の参照先

探す情報

このマニュアルに含まれる情報

Visual Basic チュートリアル

Delphi チュートリアル

Microsoft Access チュートリアル

API リファレンス マニュアル

ActiveX コントロール、OLE DB と ADO、および Pervasive Direct for Delphi and C++ Builder

Pervasive PSQL のインストールと実行

Pervasive PSQL における管理者タスクの実行

Pervasive PSQL エラー コード

参照先

「[このマニュアルの構成](#)」

「[Microsoft Visual Basic での Pervasive データの使用](#)」

「[Pervasive PSQL を Visual Basic で使用](#)」

「[Pervasive PSQL を Delphi で使用](#)」

「[Microsoft Access での Pervasive データの使用](#)」

『[Btrieve API Guide](#)』を参照してください。

『[Pervasive PSQL Programmer's Guide](#)』を参照してください。

『[Getting Started with Pervasive PSQL](#)』を参照してください。

『[Getting Started with Pervasive PSQL](#)』を参照してください。

『[Status Codes and Messages](#)』を参照してください。

このマニュアルの読者

このマニュアルは、Microsoft Access、Visual Basic、ActiveX、および Borland Delphi のユーザーを対象としており、読者が各アプリケーションの使用方を理解していることを前提としています。

このマニュアルの構成

- 第1章「[Pervasive SDK コード サンプルの概要](#)」

この章では、このチュートリアルおよび基本的な概念について説明します。
 - 第2章「[Microsoft Access での Pervasive データの使用](#)」

この章では、Microsoft Access を使用してサンプル データベース内のデータを表示および操作する方法について説明します。
 - 第3章「[Microsoft Visual Basic での Pervasive データの使用](#)」

この章では、Microsoft Visual Basic を使用してサンプル データベース内のデータを表示および操作する方法について説明します。
 - 第4章「[Pervasive PSQL を Delphi で使用](#)」

この章では、Delphi で ActiveX インターフェイスを使用して Pervasive データを表示および操作する方法について説明します。また、Btrieve API を使用したファイル オペレーション、ODBC を使用したデータベースへの接続、SQL クエリの実行方法について説明します。
 - 第5章「[Pervasive PSQL を Visual Basic で使用](#)」

この章では、Microsoft Visual Basic で ActiveX インターフェイスを使用して Pervasive データを表示および操作する方法について説明します。また、Btrieve API を使用したファイル オペレーション、ODBC を使用したデータベースへの接続、SQL クエリの実行方法について説明します。
 - 第6章「[ActiveX を使用する Pervasive PSQL アプリケーションの作成](#)」

この章では、Pervasive PSQL および Pervasive ActiveX インターフェイスを使用してアプリケーションを開発する方法について説明します。
 - 第7章「[Btrieve API の言語インターフェイス](#)」

この章には、さまざまなプログラミング インターフェイスでの Btrieve API のサンプルが用意されています。
- このマニュアルの巻末には索引が用意されています。

表記上の規則

特段の記述がない限り、コマンド構文、コード、およびコード例では、以下の表記が使用されます。

大文字小文字の 区別	通常、コマンドと予約語は、大文字で表記されます。本書で別途記述がない限り、これらの項目は大文字、小文字、あるいはその両方を使って入力できます。たとえば、MYPROG、myprog、またはMYprogと入力することができます。
太字	太字で表示される単語には次のようなものがあります。メニュー名、ダイアログ ボックス名、コマンド、オプション、ボタン、ステートメントなど。
固定幅フォント	固定幅フォントは、コマンド構文など、ユーザーが入力するテキストに使われます。
[]	省略可能な情報には、[<i>log_name</i>] のように、角かっこが使用されます。角かっこで囲まれていない情報は必ず指定する必要があります。
	縦棒は、[<i>file name</i> @ <i>file name</i>] のように、入力する情報の選択肢を表します。
< >	< > は、/D=<5 6 7> のように、必須項目に対する選択肢を表します。
変数	<i>file name</i> のように斜体で表されている語は、適切な値に置き換える必要のある変数です。
...	[<i>parameter...</i>] のように、情報の後に省略記号が続く場合は、その情報を繰り返し使用できます。
::=	記号 ::= は、ある項目が別の項目用語で定義されていることを意味します。たとえば、 <i>a</i> ::= <i>b</i> は、項目 <i>a</i> が <i>b</i> で定義されていることを意味します。

Pervasive SDK コード サンプルの概要

1

このマニュアルのチュートリアルおよびサンプル プログラムでは、Microsoft Access、ActiveX、Visual Basic、および Borland Delphi などの使い慣れたツールを使用して Pervasive データを表示および操作し、Pervasive PSQL に対応したアプリケーションを作成する方法をわかりやすく説明します。これらのチュートリアルでは、特殊クエリの実行やカスタム レポートの作成など、既存のデータにアクセスして活用する新しい方法について説明します。

Pervasive PSQL v11 またはその他のバージョンの Pervasive 製品をお使いの場合、Pervasive データを使用しています。ほかにも、Pervasive のデータは、会計用パッケージ、顧客管理システム、バックアップ用ソフトウェア、インターネット アプリケーション、生産管理システム、情報管理システムなどの各種ソフトウェア プログラムで使用されています。気付かないうちに Pervasive データをあなたのデスクトップや企業で使用していることもよくあります。

SDK コード サンプルの概要

実際の作業を始める前に、Pervasive データベース エンジンについての基本的な情報を理解する必要があります。Pervasive PSQL エンジンは、トランザクショナル アクセスおよびリレーショナル アクセスの 2 つのアクセス レイヤーから構成されています。トランザクショナル データ アクセスのパフォーマンスとリレーショナル データの堅牢性という 2 種類のアクセス 形態の統合により大きなメリットが得られます。エンジンは、スタンドアロンまたはクライアント / サーバーとして動作します。スタンドアロン環境では、エンジンは、アプリケーションやデータと同じマシン上に存在しますが、クライアント / サーバー環境では、エンジンとデータはネットワーク サーバー上に存在し、アプリケーションはクライアント マシンに存在します。

サーバー エンジンと通信してクライアントのアプリケーションでデータベースに接続するには、リクエスターを使用する必要があります (スタンドアロン環境では、リクエスターは不要)。

Pervasive の用語については、Pervasive PSQL マニュアル集に含まれている「用語集」を参照してください。

Pervasive PSQL SDK に含まれるサンプル

Pervasive PSQL v11 SDK では以下のサンプルが Web ダウンロードにより使用可能です。

Video Store アプリケーション

ビデオ販売店管理アプリケーションのサンプルが以下の言語で提供されています。

Windows

以下のフォーム ベースのサンプルが使用できます。

- Visual Basic
- Borland Delphi
- Borland C++ Builder
- Microsoft Visual C++
- Java クラス ライブラリ

以下の Web ベースのサンプルが使用できます。

- ASP
- ASP.Net
- Perl
- PHP
- JSP

Linux

- Perl
- PHP
- Java Server Pages (JSP)

これらのサンプルのソース コードは、`/usr/local/psql/sdk/samples/web` ディレクトリ内にあります。

Pervasive エンジンへのアクセス

Distributed Tuning Objects

Windows 用の Distributed Tuning Objects の使用法を示す Visual Basic のアプリケーションのサンプルが含まれています。

Distributed Tuning Objects の詳細については、『Distributed Tuning Objects Guide』を参照してください。Pervasive PSQL ファイルのデフォルトの保存場所については、『Getting Started with Pervasive PSQL』の「[Pervasive PSQL ファイルがインストールされる場所](#)」を参照してください。

サンプルのソースは、次のディレクトリを参照してください。

`file_path¥PSQL¥SDK¥SAMPLES¥DTI¥VBASIC` (`file_path` はデフォルトで `¥Program Files¥Pervasive Software` です。)

Distributed Tuning Interface (Windows 用)

以下の DTI サンプルが使用できます。

Distributed Tuning Interface の使用法を示す 2 つのアプリケーションのサンプルが含まれています。Distributed Tuning Interface の詳細については、『Distributed Tuning Interface Guide』を参照してください。

Visual C++

サンプルのソースは、次のディレクトリを参照してください。

`file_path¥PSQL¥SDK¥SAMPLES¥DTI¥MSVC` (`file_path` はデフォルトで `¥Program Files¥Pervasive Software` です。)

Delphi

サンプルのソースは、次のディレクトリを参照してください。

`file_path¥PSQL¥SDK¥SAMPLES¥DTI¥DELPHI5` (`file_path` はデフォルトで `¥Program Files¥Pervasive Software` です。)

Distributed Tuning Interface (Linux 用)

Linux 用の DTI サンプルでは以下のような操作を行います。

- ライセンスの追加または削除
- MicroKernel の使用情報の表示
- DSN およびデータベース名の表示
- 新しいデータベース、DSN およびデータベース名の作成と削除

Microsoft Access での Pervasive データの使用

2

この章では、Pervasive PSQL と共にインストールされるサンプル データベースのデータを Microsoft Access を使用して表示および操作する方法について説明します。Access を使用して Pervasive のデータに接続する場合は、ODBC インターフェイスを使用してください。

Pervasive PSQL は、Pervasive のエンジンと ODBC 対応アプリケーションとの間での通信を可能にするものです。ODBC はデータベース要求そのものは処理しません。その代わりに、ODBC 対応アプリケーション (Access など) からの要求を受け付け、その要求を Pervasive PSQL 呼び出しに変換します。

この章では、以下の項目について説明します。

- 「[レッスン 1 : 環境を設定する](#)」
- 「[レッスン 2 : テーブルにリンクする](#)」
- 「[レッスン 3 : テーブルをインポートする](#)」
- 「[レッスン 4 : SQL パススルー クエリを作成する](#)」



メモ この章のレッスンでは、Access 97 の画面を示しますが、Access 2000 でも手順はほとんど同じです。

レッスン 1 : 環境を設定する

このセクションでは、ODBC 対応アプリケーション（Microsoft Access など）で Pervasive のデータに接続するために必要な作業について説明します。作業項目は以下のとおりです。

- 「必要なソフトウェアをインストールする」
- 「サンプルデータベースをコピーする」
- 「新規データベース名および DSN を作成する」
- 「データ辞書ファイルを作成する」
- 「チュートリアル用のデータベースを作成する」

必要なソフトウェアをインストールする

以下のソフトウェアをあらかじめインストールしておく必要があります。

- Pervasive PSQL データベース エンジンまたはリクエスター。ローカル エンジン、またはクライアント リクエスターとサーバー エンジンとの組み合わせです。

これらのコンポーネントのインストール方法については、『Getting Started with Pervasive PSQL』を参照してください。

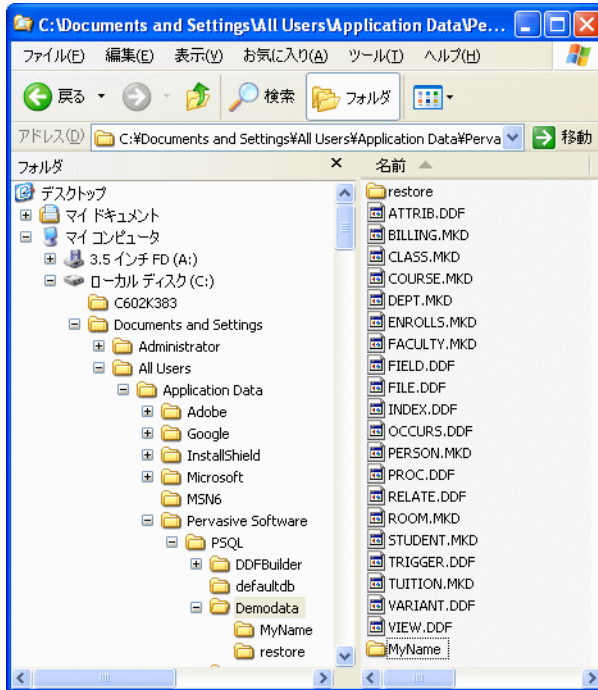
サンプル データベースをコピーする

このチュートリアルでは、Pervasive PSQL と一緒にインストールされるサンプル データベースを使用します。以下の手順に従うことにより、サンプル データベースの個人用コピーを作成できるため、デフォルトでインストールされたデータベースを保持したまま自由に変更を加えることができます。

➤ サンプル データベースをコピーするには

- 1 Pervasive PSQL と一緒にインストールされたサンプル データベースを検索します。通常、このサンプル データベースは、Pervasive PSQL サーバーの Application Data（または ProgramData）フォルダー下にある ¥PSQL¥Demodata ディレクトリにあります。

- Demodata ディレクトリに新しいフォルダーを作成してネットワークで使用するユーザー名を付けます。



- Demodata フォルダ内のファイルをすべて選択し、新しいフォルダにコピーします。このチュートリアルでは、以降この新しいフォルダを使用します。

新規データベース名および DSN を作成する

データ ソースは、アクセスするデータベースや使用する ODBC ドライバーをはじめ、その他接続に関連する情報を指定します。データ ソースは、アクセスする各データベースごとに、ODBC アドミニストレーターを使用して定義する必要があります。

➤ データ ソースとデータベース名を定義するには

- [スタート]メニューから Pervasive プログラム グループにアクセスし、Pervasive PSQL Control Center を起動します。
- Pervasive PSQL エクスプローラーの " エンジン " ノードを展開して、Pervasive PSQL Control Center に登録されているデータベース エンジンを表示します。

- 3 希望するデータベース エンジンが見つからない場合は、" エンジン " ノードを右クリックして [新規作成 | サーバー] の順にクリックし、ネットワーク上にあるデータベース サーバーを選択します。そうでない場合は、次の手順に進みます。
- 4 作業対象のデータベース エンジンのアイコンを右クリックして、[新規作成 | データベース] の順にクリックします。
メモ : 別のマシンにあるサーバー エンジンを使用している場合、新規データベースを作成するためには、そのサーバーのオペレーティングシステムでの管理者権限が必要です。
- 5 ウィザードの指示に従って、データベースを作成します。

データ辞書ファイルを作成する

.ddf ファイルがない、または .ddf ファイルが空の Btrieve データを使用している場合は、このセクションを参照してください。

データ辞書は、テーブル、列、インデックス (Btrieve では、ファイル、フィールド、キーという用語を用います) を使用して Btrieve データを説明する .ddf ファイルの集まりです。Btrieve データベースには、そのデータベース内のデータの形式や意味を説明する情報はありますが、この情報は Btrieve アプリケーションに定義されます。

データ辞書は、データベースのコンポーネントを定義して、ODBC、Pervasive PSQL をはじめ、その他の各種ツールおよびアプリケーションから Btrieve データベースにアクセスすることを可能にします。最低限必要な .ddf ファイルは、File.ddf、Field.ddf、Index.ddf で、これらのファイルにより、Access からデータベースに接続することが可能になります。

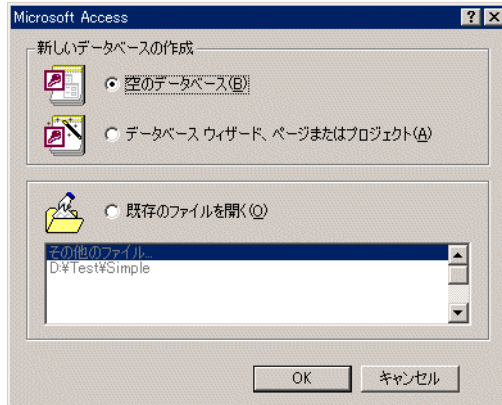
DDF ファイルは、Access で使用するために、データベースを正確に記述する必要があります。このため、Pervasive には、データベースのテーブル、列、インデックスを定義するツールとして Pervasive PSQL Control Center が用意されています。詳細については、Pervasive PSQL User's Guide を参照してください。

チュートリアル用のデータベースを作成する

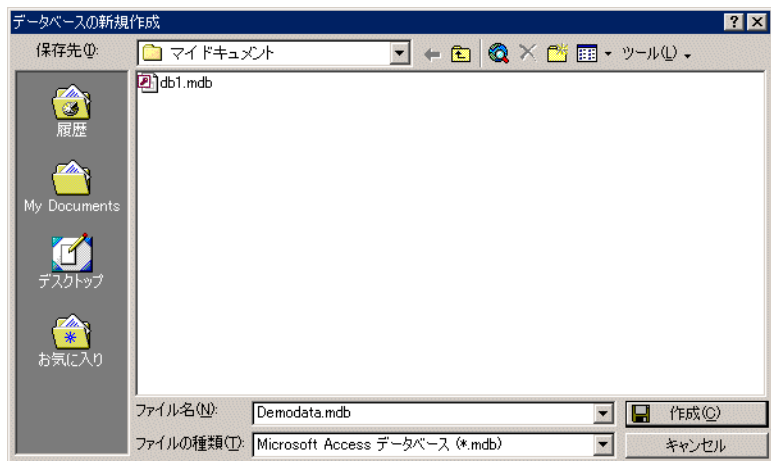
ここでは、新しいデータベースを Access で作成する手順を説明します。このデータベースは以降のレッスンで使用することができます。

➤ Access でチュートリアル用のデータベースを作成するには

- 1 Microsoft Access を起動し、空のデータベースを新しく作成します。



- 2 [データベースの新規作成] ダイアログ ボックスの [ファイル名] フィールドに「Demodata.mdb」と入力し、[作成] をクリックします。



- 3 「Demodata」という名前の空のデータベースが新たに作成されます。

レッスン 2 : テーブルにリンクする

Microsoft Access では、ODBC 経由で Pervasive のデータベースおよびそのコンポーネントが認識されます。Access で Pervasive のデータを表示および操作する方法には、リンク テーブルの作成があります。リンク テーブルを作成した場合、そのデータは、現在の場所にそのままの形式で残ります。リンクすることにより、最新のデータにアクセスすることが可能になり、ほかのアプリケーションでも引き続きデータの表示および更新が可能です。

このレッスンでは、サンプル データベース内の各テーブルにリンクします。

➤ サンプル データベースへのリンク テーブルを作成するには

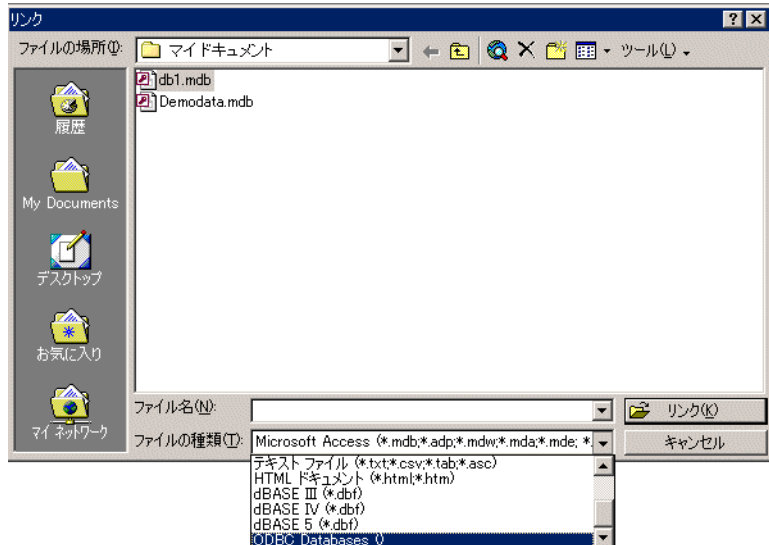
- 1 Access 2000 では、次の手順に従います。[Demodata: データベース] ウィンドウのテーブルを右クリックし、[テーブルのリンク] を選択します。手順 3 に進みます。

Access 97 では、次の手順に従います。[Demodata: データベース] ウィンドウの [テーブル] タブをクリックし、[新規作成] をクリックします。次の手順に進みます。

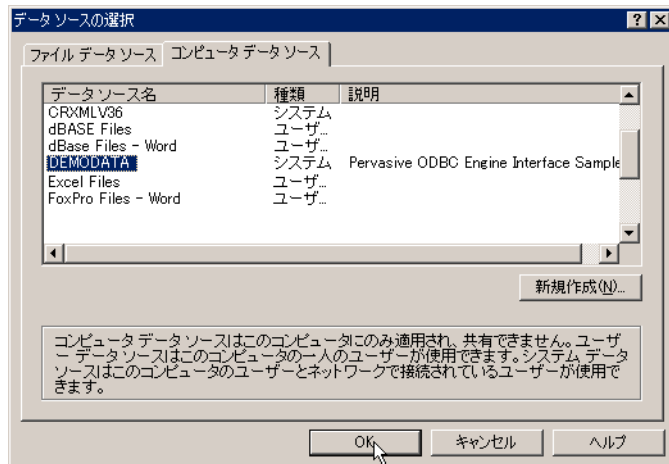
- 2 [テーブルの新規作成] ダイアログ ボックスで [テーブルのリンク] を選択し、[OK] をクリックします。



- 3 [リンク] ダイアログ ボックスの下部にある [ファイルの種類] から "ODBC Databases" を選択します。

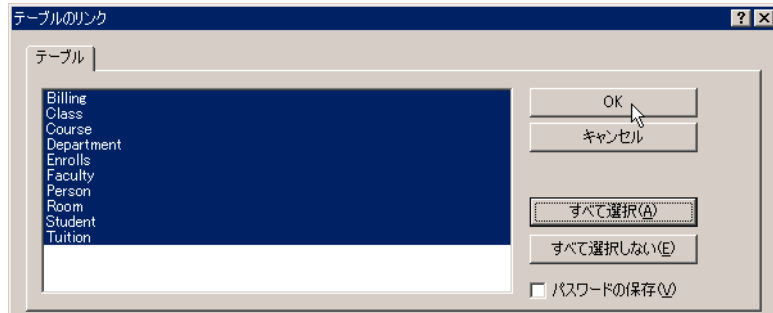


- 4 [データソースの選択] ダイアログボックスの [コンピューターデータソース] タブをクリックし、Demodata データソースを選択して [OK] をクリックします。

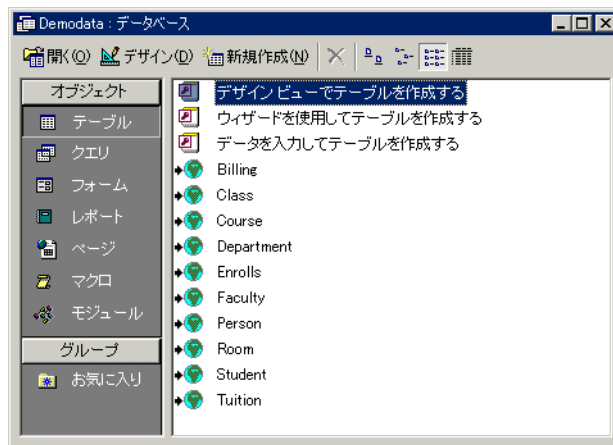


メモ データソースは、必ず [コンピューターデータソース] タブから選択してください。誤って [ファイルデータソース] タブのデータソースを選択した場合は、予期しない結果が生じることがあります。

- 5 [テーブルのリンク] ダイアログボックスで、[すべて選択] をクリックし、次に [OK] をクリックします。



Pervasive のサンプル データベースに、すべての既存テーブルに対するリンク テーブルが作成されます。



これらのテーブルのデータは、Access のほかのテーブルと同じように表示および操作することができ、行を挿入してデータを更新することも可能です。また、Access でのフィールド表示を変更することは可能ですが、リンク テーブルおよびそのフィールドのソース データベースでの定義を変更することはできません。ソース データベースの変更には、SQL パススルー クエリを使用して DDF を変更する必要があります。その方法については、「[レッスン 4 : SQL パススルー クエリを作成する](#)」で説明します。

ODBC アドミニストレーターを使用してデータ ソースの設定を変更した場合は、テーブルを再リンクしてください。

また、Access に搭載されている Jet エンジンでは、ODBC からデータに接続する際に、いくつかの制限があります。詳細については、Microsoft の「[Tech Notes Q128809](#)」および「[Tech Notes Q127096](#)」を参照してください。

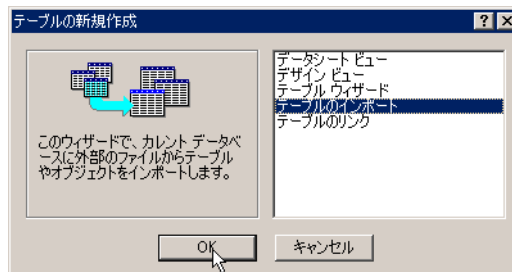
レッスン 3 : テーブルをインポートする

Microsoft Access では、ODBC 経由で Pervasive のデータベースおよびそのコンポーネントが認識されます。Pervasive のデータを Access で表示および操作するためのもう 1 つの方法は、インポート テーブルを作成することです。インポート テーブルを作成した場合は、Access にデータのコピーが作成され、ソース データは更新されません。つまり、インポートしたテーブルに変更を加えた場合、Access ではその変更を確認できますが、ソースである Pervasive データに接続したほかのユーザーは、それを確認できません。インポートは、Access に取り込んだデータを Access のみで使用する場合には便利ですが、Access 以外のアプリケーションでも使用する場合は、リンクを選択することをお勧めします。

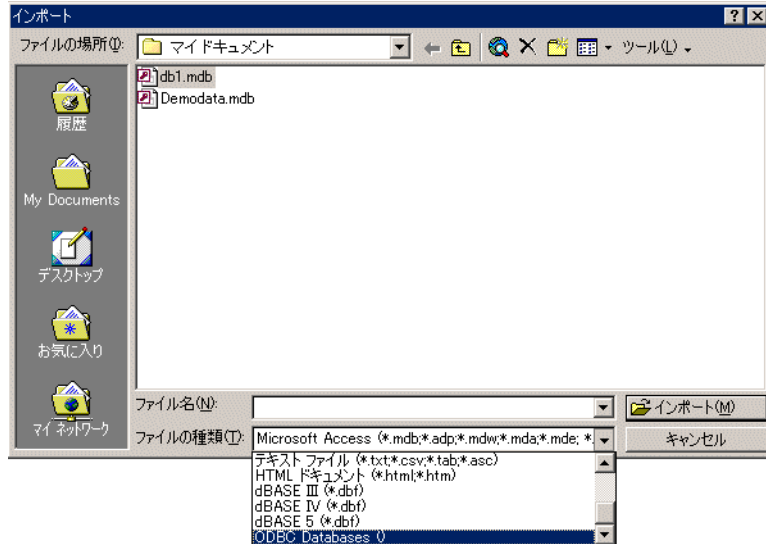
このレッスンでは、サンプル データベースからすべてのテーブルをインポートします。

➤ サンプル データベースからテーブルをインポートするには

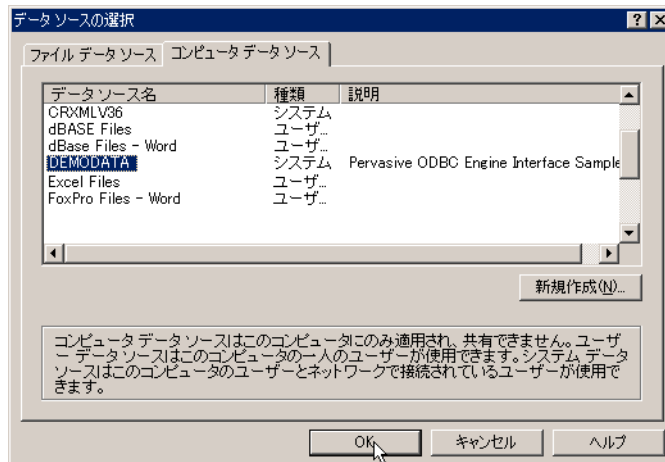
- 1 [Demodata: データベース] ウィンドウの [テーブル] タブをクリックし、[新規作成] をクリックします。
- 2 [テーブルの新規作成] ダイアログ ボックスで [テーブルのインポート] を選択し、[OK] をクリックします。



- 3 [インポート] ダイアログ ボックスで、[ファイルの種類] フィールドから "ODBC Databases" を選択します。



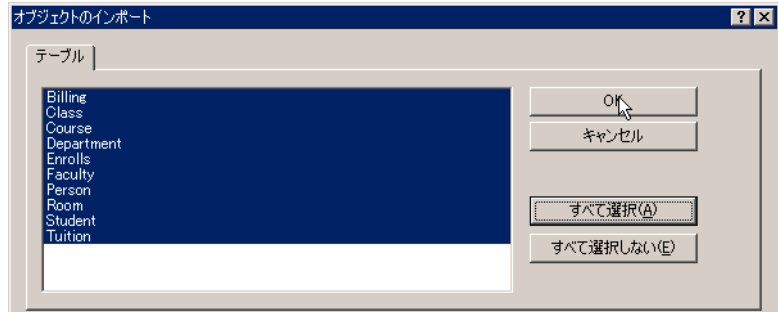
- 4 [データ ソースの選択] ダイアログ ボックスの [コンピューター データ ソース] タブをクリックし、Demodata データ ソースを選択して [OK] をクリックします。



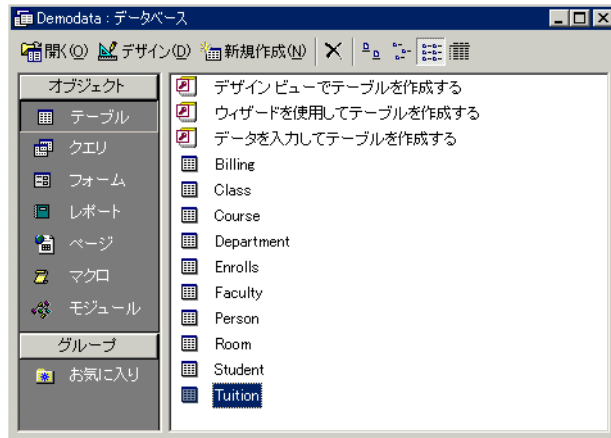
メモ データ ソースは、必ず [コンピューター データ ソース] タブから選択してください。誤って [ファイル データ ソース] タブのデータ ソースを選択した場合は、予期しない結果が生じることがあります。

レッスン 3 : テーブルをインポートする

- 5 [オブジェクトのインポート] ダイアログ ボックスで、[すべて選択] をクリックし、[OK] をクリックします。



Pervasive サンプル データベースに、すべての既存テーブルに対するインポート テーブルが作成されます。



インポート テーブルのアイコンは、リンク テーブルのアイコンとは異なります。

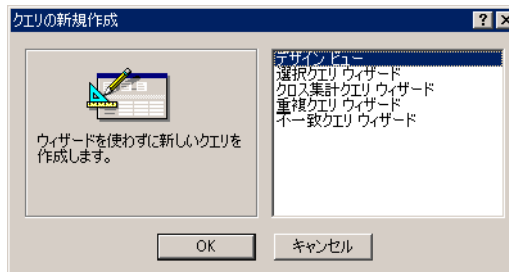
レッスン 4 : SQL パススルー クエリを作成する

Microsoft Access では、SQL ステートメントを使用してクエリを作成することができます。通常、クエリを作成する際は、SQL ステートメントが自動的に処理されます。パススルー クエリを作成した場合、SQL ステートメントは変更されないまま ODBC ドライバーに渡されます。次に、クエリは ODBC ドライバーにより Pervasive の ODBC インターフェイスに渡され、ODBC リクエストが Pervasive エンジン リクエストに変換されます。

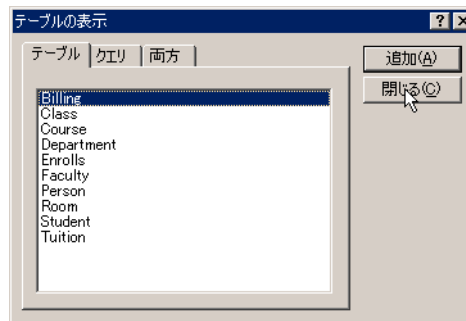
このレッスンでは、Pervasive サンプル データベースの Course テーブルのデータを表示する簡単なクエリを作成します。

➤SQL パススルー クエリを作成するには

- 1 [Demodata: データベース] ウィンドウの [クエリ] タブで [新規作成] をクリックします。
- 2 [クエリの新規作成] ダイアログ ボックスで [デザインビュー] を選択し、[OK] をクリックします。



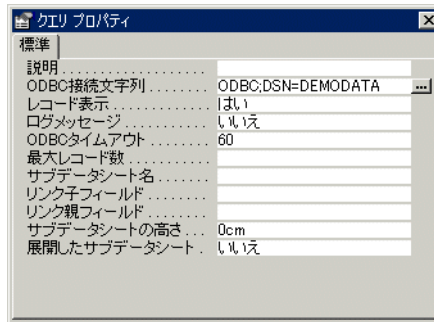
- 3 [テーブルの表示] ダイアログ ボックスの [閉じる] をクリックします。



デフォルトでは、新しいクエリを作成する基となる既存のテーブルまたはクエリを選択するよう指示されますが、ここでは SQL パススルークエリを作成しているため、このダイアログ ボックスを使用する必要はありません。[閉じる] をクリックします。

レッスン 4 : SQL パススルー クエリを作成する

- Access のメイン メニューから [クエリ | SQL | パス スルー] を選択します。
- Access のメイン メニューから、[表示 | プロパティ] を選択します。
- [クエリ プロパティ] ダイアログ ボックスの [ODBC 接続文字列] フィールドに「DSN=Demodata」を追加し、ダイアログ ボックスを閉じます。



この文字列によりレッスン 1 で作成したデータ ソース Demodata が認識されます。このクエリのプロパティを設定することにより、クエリと Demodata が関連付けられます。プロパティを設定しない場合は、クエリを実行するたびにデータ ソース名を入力するよう指示されます。

- [SQL パススルー クエリ] ウィンドウに、以下の SQL ステートメントを入力します。

```
SELECT * FROM Course
```

Access では、ステートメントの最後に使用される Pervasive PSQL v11 の標準的な区切り文字が正しく処理されません。ステートメントの区切り文字としてセミコロン (;)、シャープ (#)、円記号 (¥)、逆引用符 (‘) などを使用した場合は、ステータス コード 501 が表示されます。

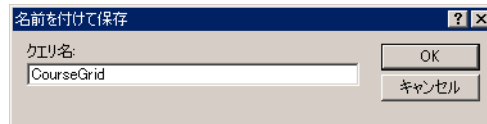
作成する SQL ステートメントは、Pervasive PSQL v11 構文の条件に準拠している必要があります。基本的な構文の説明については、『SQL Engine Reference』を参照してください。

- 8 メイン メニューの [クエリ] から、[実行] を選択します。クエリの結果が表示されます。



Name	Description	Credit_Hours	Dept_Name
ACC 101	Accounting I	3	Accounting
ACC 102	Accounting II	3	Accounting
ACC 203	Intermediate Acco	3	Accounting
ACC 204	Intermediate Acco	3	Accounting
ACC 305	Taxation	3	Accounting
ACC 306	Cost Accounting	3	Accounting
ACC 407	Advanced Accour	3	Accounting
ACC 408	Auditing	4	Accounting
ANT 101	Cultural Anthropol	3	Anthropology
ANT 202	Development of E	3	Anthropology
ANT 203	Social Change	3	Anthropology
ANT 304	Ethnography	3	Anthropology
ANT 405	Independent Stud	3	Anthropology

- 9 [SQL パススルー クエリ] ウィンドウを閉じ、クエリに「CourseGrid」と名前を付けて保存します。



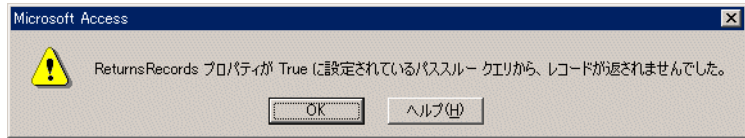
SQL パススルー クエリを使用することにより、Access のテーブルのデザインビューでは実行不可能なタスクを実行できます。たとえば、SQL パススルー クエリを使用してテーブルに列を追加することができます。

以下のセクションでは、SQL ステートメントの例をいくつか説明します。詳細については、『SQL Engine Reference』を参照してください。SQL ステートメントの例を [SQL パススルー] ウィンドウにコピーした場合は、構文エラーを防ぐために一部の特殊文字（引用符など）を再入力する必要があります。

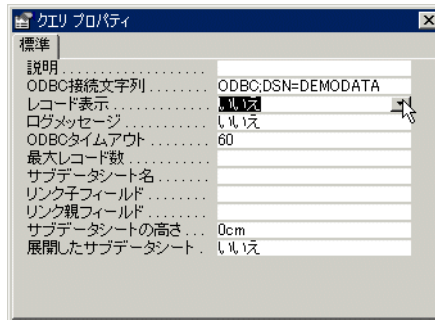
Access における SQL ステートメント作成を許可する場合（デザインビューでクエリを作成する場合など）、そのステートメントは Pervasive PSQL v11 構文ではなく、Jet SQL 構文に適合している必要があります。したがって、Access で作成された SQL ステートメントは、Pervasive PSQL v11 の構文エラーになる可能性があります。Access のデフォルトでは、SQL ステートメントは Jet エンジンで処理されることが前提であるため、このエラーが発生しますが、SQL パススルー クエリを使用する場合、ステートメントは Jet エンジンを回避し、処理されることなく Pervasive PSQL v11 に渡されます。

レッスン 4 : SQL パススルー クエリを作成する

最後に、以下のステートメントはレコードを返さないことに注意してください。デフォルトのクエリ プロパティをお使いの場合は、次の警告メッセージが表示されます。



この警告が結果に影響を及ぼすことはありませんが、このメッセージを表示させないようにすることはできます。デザイン ビューでクエリを開き、ツールバーの [プロパティ] ボタンをクリックします。次のように、[クエリ プロパティ] の [レコード表示] フィールドを [いいえ] に設定します。



ALTER TABLE ステートメント

以下のステートメントは、Department テーブルに緊急連絡先の電話番号を保存する列を追加します。

```
ALTER TABLE Department ADD Emergency_Phone CHAR(20)
```

更新された Department テーブルを確認するには、Access に再リンクします。

INSERT ステートメント

次のステートメントは、Course テーブルに行を 1 行挿入します。

```
INSERT INTO Course(Name, Description, Credit_Hours, Dept_Name)
VALUES('CHE 308', 'Organic Chemistry II', 4, 'Chemistry')
```

UPDATE ステートメント

次のステートメントは、Course テーブルにある ECO 305 の単位時間を 3 から 4 に変更します。

```
UPDATE Course SET Credit_Hours = 4 WHERE Name = 'ECO 305'
```

まとめ

作業	操作手順
環境の設定	<ul style="list-style-type: none">◆ Pervasive PSQL Control Center を使用してデータソースを定義する◆ 必要に応じて .ddf ファイルを定義する
リンクテーブル、またはインポート テーブルを作成する	<ul style="list-style-type: none">◆ [テーブル] タブの [新規作成] をクリックする◆ ODBC データベースをリンクまたはインポートする◆ [コンピューター データ ソース] タブでデータソースを選択する
SQL パススルー クエリを実行する	<ul style="list-style-type: none">◆ [クエリ] タブの [新規作成] をクリックする◆ メイン メニューから、[クエリ SQL パススルー] を選択する◆ クエリのプロパティ シートの [ODBC 接続文字列] フィールドを設定する (DSN = DSN 名)◆ クエリがレコードを 1 件も返さない場合は、[クエリ プロパティ] の [レコード表示] フィールドを "いいえ" に設定する

Microsoft Visual Basic での Pervasive データの使用

3

この章では、Pervasive PSQL と共にインストールされるサンプル データベースのデータを Microsoft Visual Basic を使用して表示および操作する方法について説明します。Visual Basic を使用して Pervasive のデータにアクセスする場合、以下の方法から選択できます。

- Visual Basic のバウンド ActiveX Data Objects (ADO)
- Classic Software などのサード パーティ製データ コントロール
- 直接 ODBC API 呼び出し
- 直接 Btrieve API 呼び出し

これらの方法は、開発時間の長さや実行時のパフォーマンスの速さの順に記載されており、開発時間は、ADO アプリケーションが最短、直接 Btrieve アプリケーションが最長です。逆に、ランタイム時のパフォーマンスは、直接 Btrieve アプリケーションが最速で、ADO アプリケーションが最も遅くなります。

この章では、以下の項目について説明します。

- 「[レッスン 1：環境を設定する](#)」
- 「[レッスン 2：ADO データ コントロール \(ADODC\) を使用する](#)」
- 「[レッスン 3：ActiveX Data Object でコードを記述する](#)」

サード パーティ製コントロール使用の詳細については、その製品の製造元にお問い合わせください。直接 ODBC API 呼び出しの使用の詳細については、『SQL Engine Reference』を参照してください。直接 Btrieve API 呼び出しの使用の詳細については、『Btrieve API Guide』を参照してください。



メモ この章のレッスンで使用している図は、Visual Basic 6 の画面です。必要な場合は、Visual Basic 5 と 6 の違いを説明します。Visual Basic 3 でもバウンド テキスト ボックス データ コントロールは使用できますが、このチュートリアルでは、Visual Basic 3 については説明しません。

レッスン 1 : 環境を設定する

このセクションでは、ODBC 対応アプリケーション (Microsoft Visual Basic など) で Pervasive のデータに接続するために必要な作業について説明します。作業項目は以下のとおりです。

- 「必要なソフトウェアをインストールする」
- 「サンプルデータベースをコピーする」
- 「新規データベース名および DSN を作成する」
- 「データ辞書ファイルを作成する」
- 「チュートリアル用のプロジェクトを作成する」

必要なソフトウェアをインストールする

以下のソフトウェアをあらかじめインストールしておく必要があります。

- Pervasive PSQL データベース エンジンまたはリクエスター。ローカル エンジン、またはクライアント リクエスターとサーバー エンジンとの組み合わせです。

これらのコンポーネントのインストール方法については、『Getting Started with Pervasive PSQL』を参照してください。

サンプル データベースをコピーする

このチュートリアルでは、Pervasive PSQL と一緒にインストールされるサンプル データベースを使用します。以下の手順に従うことにより、サンプル データベースの個人用コピーを作成できるため、デフォルトでインストールされたデータベースを保持したまま自由に変更を加えることができます。

➤ サンプル データベースをコピーするには

- 1 Pervasive PSQL と一緒にインストールされたサンプル データベースを検索します。通常、このサンプル データベースは、Pervasive PSQL サーバーの Application Data (または ProgramData) フォルダー下にある ¥PSQL¥Demodata ディレクトリにあります。
- 2 Demodata ディレクトリに新しいフォルダーを作成してネットワークで使用するユーザー名を付けます。
- 3 Demodata フォルダー内のファイルをすべて選択し、個人用フォルダーにコピーします。このチュートリアルでは、以降この個人用フォルダーを使用します。

新規データベース名および DSN を作成する

データ ソースは、アクセスするデータベースや、使用する ODBC ドライバーをはじめ、その他接続に関連する情報を指定します。データ ソースは、アクセスする各データベースごとに、ODBC アドミニストレーターを使用して定義する必要があります。

➤ データ ソースとデータベース名を定義するには

- 1 [スタート]メニューから Pervasive プログラム グループにアクセスし、Pervasive PSQL Control Center を起動します。
- 2 " エンジン " ノードを展開して、PCC に登録されているデータベース エンジンを表示します。
- 3 希望するデータベース エンジンが見つからない場合は、" エンジン " ノードを右クリックして [新規作成 | サーバー] の順にクリックし、ネットワーク上にあるデータベース サーバーを選択します。そうでない場合は、次の手順に進みます。
- 4 作業対象のデータベース エンジンのアイコンを右クリックして、[新規作成 | データベース] の順にクリックします。

メモ：別のマシンにあるサーバー エンジンを使用している場合、新規データベースを作成するためには、そのサーバーのオペレーティング システムでの管理者権限が必要です。

- 5 ウィザードの指示に従って、データベース名を作成します。DSN 名には、既に使用されていないければ、どのような名前でも付けることができます。

データ辞書ファイルを作成する

.ddf ファイルがない、または .ddf ファイルが空の Btrieve データを使用している場合は、このセクションを参照してください。

データ辞書は、テーブル、列、インデックス（Btrieve では、ファイル、フィールド、キーという用語を用います）を使用して Btrieve データを説明する .ddf ファイルの集まりです。Btrieve データベースには、そのデータベース内のデータの形式や意味を説明する情報はありますが、この情報は Btrieve アプリケーションに定義されます。

データ辞書は、データベースのコンポーネントを定義して、ODBC、Pervasive PSQL v11をはじめ、その他の各種ツールおよびアプリケーションから Btrieve データベースにアクセスすることを可能にします。最低限必要な .ddf ファイルは、File.ddf、Field.ddf、Index.ddf で、これらのファイルにより、Visual Basic からデータベースに接続することが可能になります。

DDF ファイルは、Visual Basic で使用するために、データベースを正確に記述する必要があります。このため、Pervasive には、データベースのテーブル、列、インデックスを定義するツールとして Pervasive PSQL Control

Center が用意されています。詳細については、Pervasive PSQL User's Guide を参照してください。

チュートリアル用のプロジェクトを作成する

ここでは、以降のレッスンで使用するための新しいプロジェクトを Visual Basic で作成する手順を紹介します。

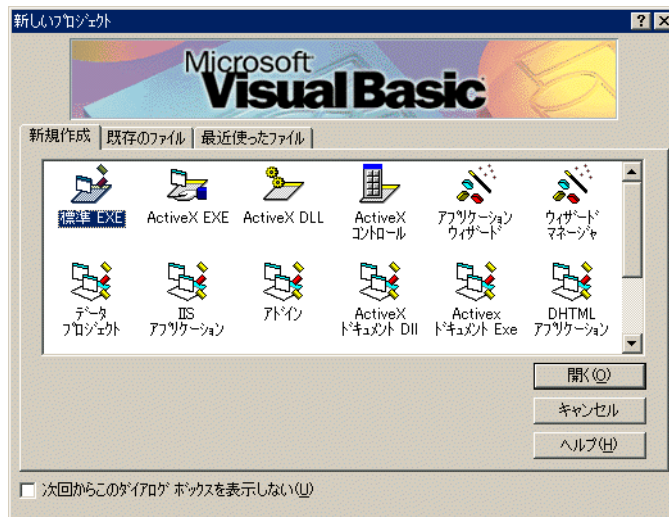
➤ Visual Basic でチュートリアル用のプロジェクトを作成するには

1 Microsoft Visual Basic を起動します。Visual Basic が、開始時に [新しいプロジェクト] ダイアログを表示するように設定されている場合は、手順 3 に進みます。

2 [ファイル] メニューの [新しいプロジェクト] をクリックします。

Visual Basic 3 または 4 をお使いの場合は、以上で終了です。次のレッスンに進んでください。Visual Basic 5 をお使いの場合は、次の手順に進みます。

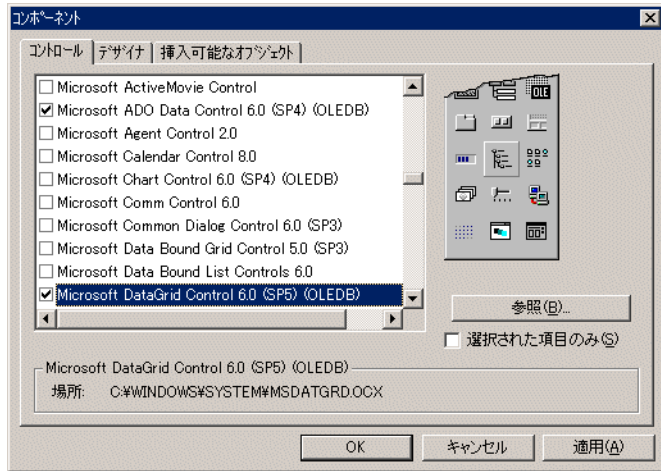
3 [新しいプロジェクト] ダイアログ ボックスで、[標準 EXE] を選択して [OK] をクリックします。



4 [ツールボックス] ウィンドウを右クリックして [コンポーネント] を選択します。

5 [コンポーネント] ダイアログ ボックスで、"Microsoft ADO Data Control 6.0 (OLEDB)" と "Microsoft DataGrid Control 6.0 (OLEDB)" を選択して [OK] をクリックします。

- 6 必要に応じてツールボックスのサイズを変更し、追加したデータバウンドコントロールを表示させます。



レッスン 2 : ADO データ コントロール (ADODC) を使用する

Microsoft ADO データ コントロール (ADODC) は、アプリケーションの作成を非常に簡単にする方法で、コードを記述することなく多くのデータ アクセスや操作を行うことができます。データ コントロールを既存のデータベースにバインドすることにより、プログラミング作業が大幅に軽減されます。

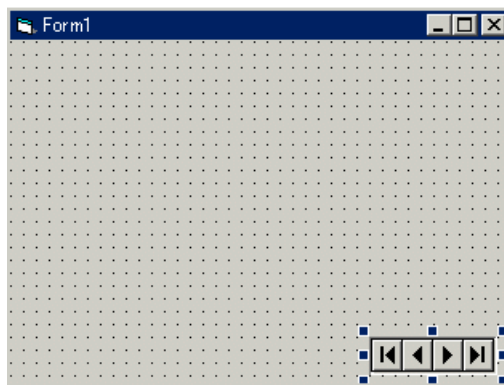
このレッスンでは、ADO データ コントロールをバウンド DataGrid コントロールにバインドして、サンプル データベースの Course テーブルのデータを表示させます。

➤Pervasive の OLE DB プロバイダーを使用してバウンド DataGrid コントロールを作成するには

- 1 ツール ボックスの ADO データ コントロールをクリックします。



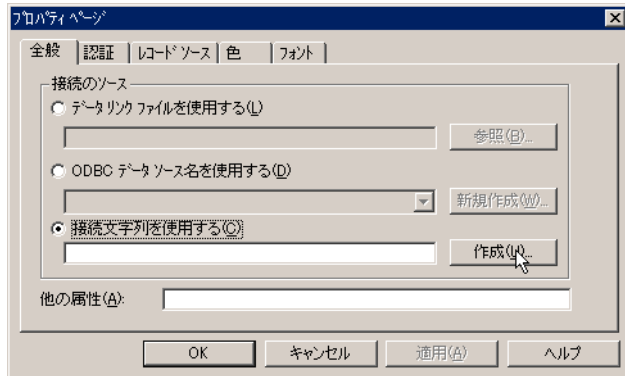
- 2 Form1 ウィンドウでカーソルをドラッグして ADO データ コントロールを作成します。



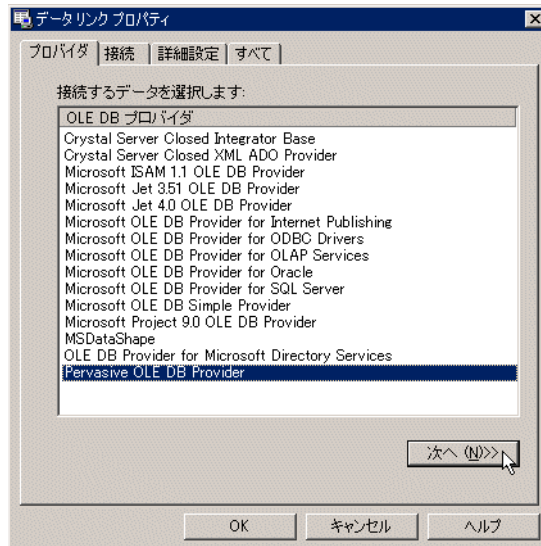
ADO データ コントロールは、データへのアクセスを可能にします。ここでは、OLE DB から Pervasive データにアクセスします。この設定は、次の手順で [プロパティ] ウィンドウを使用して指定します。

レッスン 2 : ADO データ コントロール (ADODC) を使用する

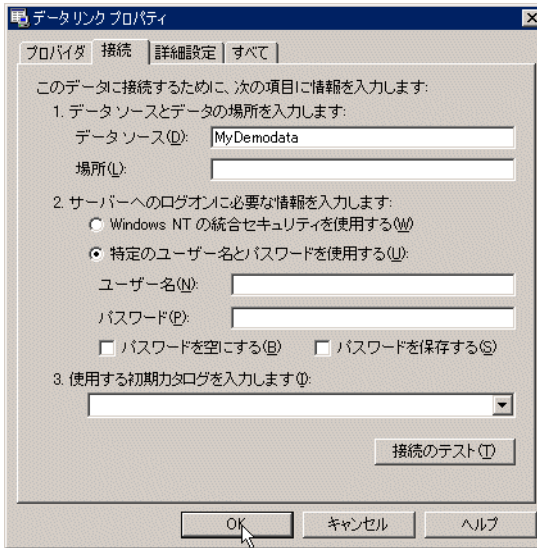
- 3 Form1 ウィンドウで ADODC コントロールを右クリックします。
[ADODC のプロパティ] を選択します。[作成] ボタンをクリックします。



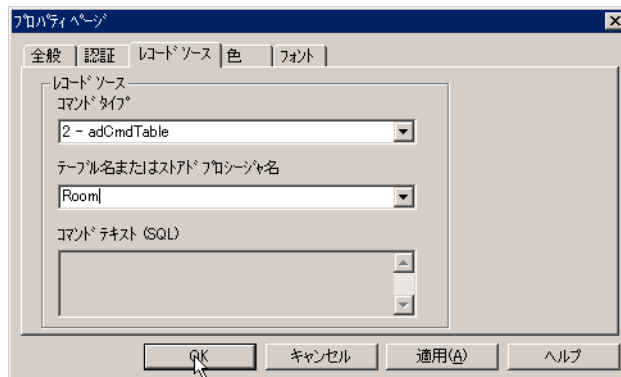
- 4 "Pervasive OLE DB Provider" を選択します。[次へ >>] ボタンをクリックします。



- 5 "MyDemodata" (またはデータベースに付けた名前) を入力します。
[OK] をクリックします。



- 6 [レコード ソース] タブをクリックします。[コマンド タイプ] を "2-adCmdTable" に変更します。[テーブル名またはストアド プロシージャ名] コンボ ボックスで "Room" を選択または入力します。[OK] をクリックします。

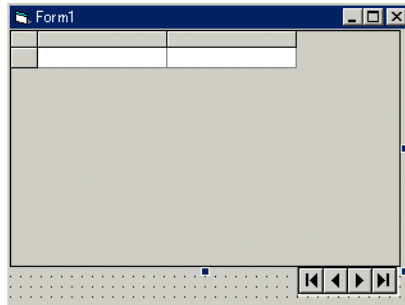


- 7 ツール ボックスの [DataGrid] コントロールをクリックします。



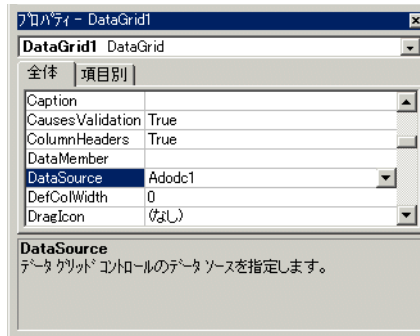
レッスン 2 : ADO データ コントロール (ADODC) を使用する

- Form1 ウィンドウでカーソルをドラッグして DataGrid コントロールを作成します。



DataGrid コントロールには、ADO データ コントロールからアクセスしたデータを表示させることができます。ここでは、手順 2 で作成した Adodc1 コントロールにアクセスします。この設定は、次の手順で [プロパティ] ウィンドウを使用して指定します。

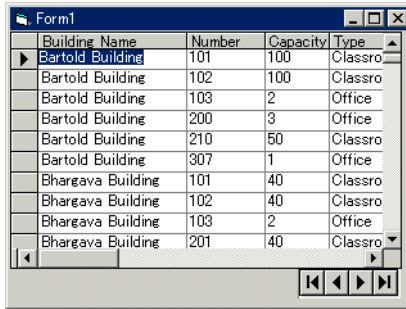
- [プロパティ] ウィンドウの [DataSource] フィールドで、"Adodc1" を選択します。



- ツール バーの [開始] をクリックします。



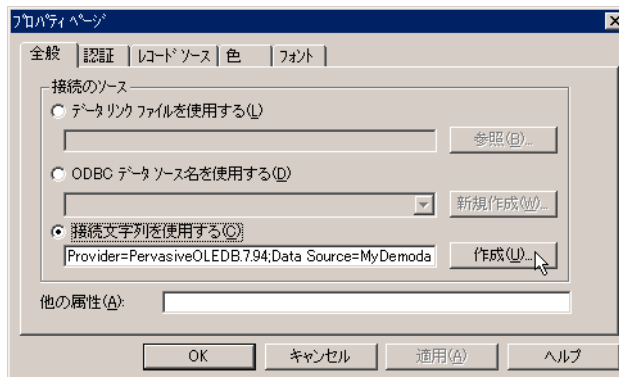
- 11 Visual Basic がアプリケーションを起動して Room テーブル内のすべてのレコードが表示されます。スクロールバー、または Adodc1 コントロールの矢印を使用することにより、行を移動できます。



- 12 レコード全体をスクロールした後、アプリケーションを終了してフォームデザインビューに戻ります。

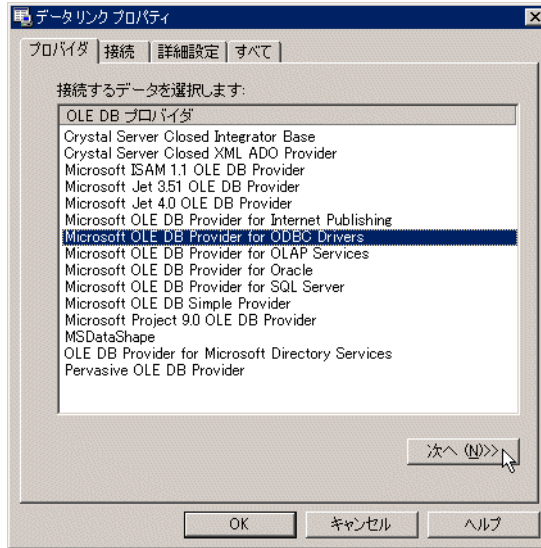
➤ Microsoft の OLE DB for ODBC Driver プロバイダーを使用してバウンド DataGrid コントロールを作成するには

- 1 Form1 ウィンドウで ADODC コントロールを右クリックします。[ADODC のプロパティ] を選択します。[作成] ボタンをクリックします。

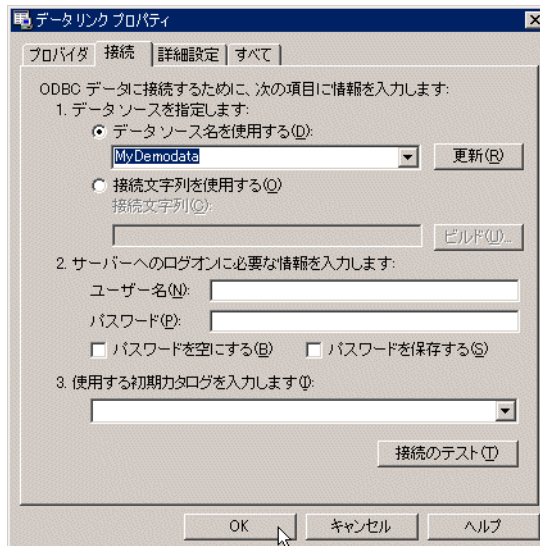


レッスン 2 : ADO データ コントロール (ADODC) を使用する

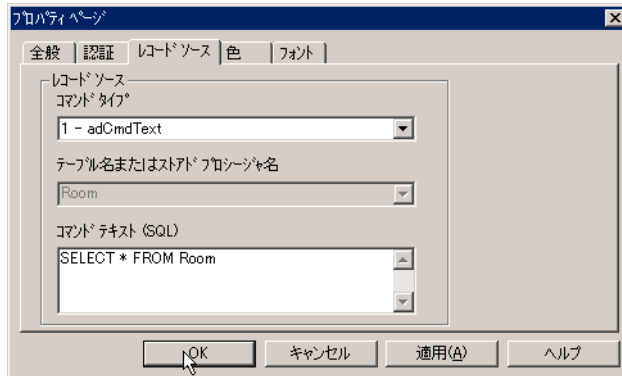
- 2 [プロバイダー] タブをクリックします。"Microsoft OLE DB Provider" を選択します。[次へ >>] ボタンをクリックします。



- 3 [データ ソース名を使用する] コンボ ボックスで "MyDemodata" (またはデータベースに付けた名前) を選択します。[OK] をクリックします。



- 4 [レコード ソース] タブをクリックします。[テーブル名またはストア プロシージャ名] コンボ ボックスで、テーブル名 "Room" を削除します。[コマンド タイプ] を "1 - adCmdText" に変更します。[コマンド テキスト (SQL)] ボックスに "SELECT * FROM Room" と入力します。[OK] をクリックします。



レッスン 3 : ActiveX Data Object でコードを記述する

ADO データ コントロールにより、データに簡単にアクセスすることができますが、数行のコードを記述することによりさらに多くのことが実行できます。たとえば、SQL クエリを実行してその結果をバウンド DataGrid コントロールに表示させることができます。

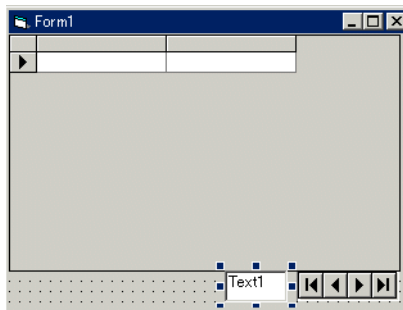
このレッスンでは、レッスン 2 で作成したバウンド DataGrid コントロールを使用して SQL ステートメントの結果を表示させます。このレッスンの終わりには、テキスト フィールドに人数を入力できるようになり、その人数以上を収容できる部屋が検出されます。このようなフォームは、ミーティングを計画する際に役立ちます。

➤ ADO から SQL ステートメントを実行するには

- 1 ツール ボックスの [TextBox] コントロールをクリックします。



- 2 Form1 ウィンドウでカーソルをドラッグして TextBox コントロールを作成します。

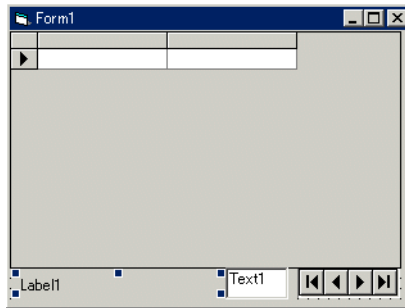


この操作により、フォームから入力できるようにするためのテキスト入力ボックスが作成されます。

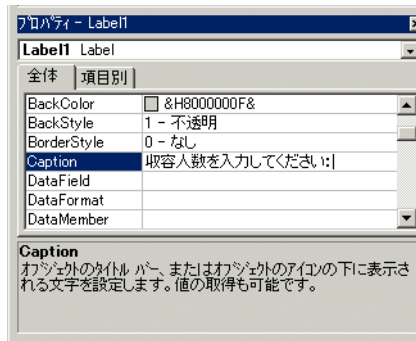
- 3 ツール ボックスの [Label] コントロールをクリックします。



- 4 Form1 ウィンドウ内でカーソルをドラッグして Label コントロールを作成します。この操作により、テキスト ボックスのラベルが作成されます。

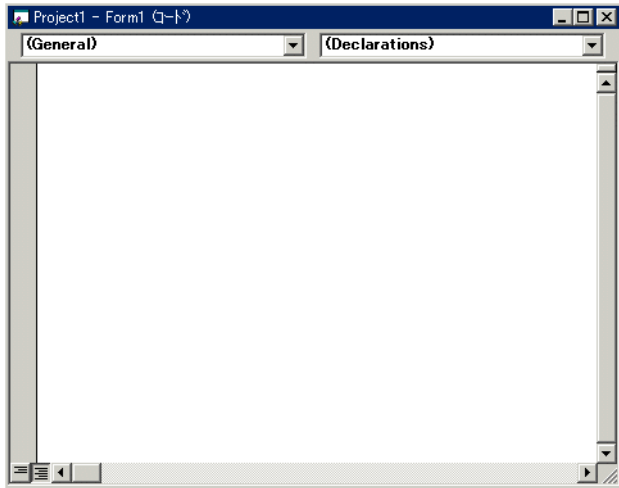


- 5 プロパティ ウィンドウの [Caption] フィールドに、" 収容人数を入力してください : " と入力します。これは、ユーザーに指示を与えるためのものです。

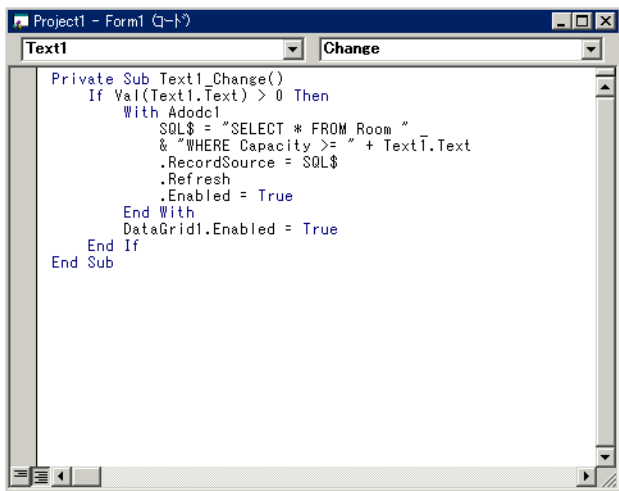


コントロールが配置されたため、表示させるレコードを取得する SQL ステートメントを記述することができます。

- 6 メイン メニューから [表示 | コード] を選択します。Visual Basic のコード ウィンドウが表示されます。

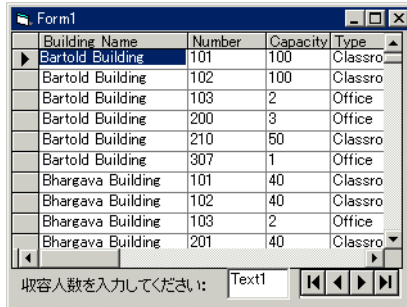


- 7 一番左のリスト ボックスで, "Text1" オブジェクトを選択します。Text1_Change と呼ばれる空の Sub プロシージャが作成されます。次のコードを入力して SQL ステートメントを作成します。



Microsoft Visual Basic での Pervasive データの使用

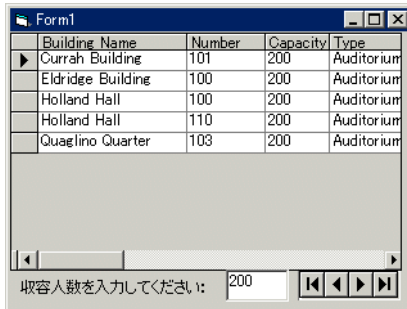
- 8 ツール バーの [開始] ボタンをクリックします。Visual Basic がアプリケーションを起動して結果を表示します。



Building Name	Number	Capacity	Type
Bartold Building	101	100	Classro
Bartold Building	102	100	Classro
Bartold Building	103	2	Office
Bartold Building	200	3	Office
Bartold Building	210	50	Classro
Bartold Building	307	1	Office
Bhargava Building	101	40	Classro
Bhargava Building	102	40	Classro
Bhargava Building	103	2	Office
Bhargava Building	201	40	Classro

収容人数を入力してください: Text1

- 9 作成したアプリケーションをテストするため、テキスト ボックスに値を入力します。DBGrid 内の結果を確認します。



Building Name	Number	Capacity	Type
Currah Building	101	200	Auditorium
Eldridge Building	100	200	Auditorium
Holland Hall	100	200	Auditorium
Holland Hall	110	200	Auditorium
Quaglino Quarter	103	200	Auditorium

収容人数を入力してください: 200

テキスト ボックスの値を変更した場合は、SQL ステートメントが Microsoft Jet エンジンにより解析され、ODBC リクエストが発行されて Pervasive PSQL エンジンに渡されます。SQL ステートメントは Microsoft Jet エンジンによって解析されるため、アプリケーションの SQL ステートメントは、Jet の SQL 構文に準拠している必要があります。この構文の詳細については、Visual Basic のヘルプを参照してください。

- 10 アプリケーションのテスト後、Form1 ウィンドウを閉じ、フォーム デザイン ビューに戻ります。

まとめ

作業	操作手順
環境の設定	<ul style="list-style-type: none"> ◆ Pervasive PSQL Control Center を使用してデータソースを定義する ◆ 必要に応じて .ddf ファイルを定義する
バウンド ADO データ コントロールを作成する	<ul style="list-style-type: none"> ◆ ADO データ コントロールを作成する ◆ データ コントロールの [レコードソース] プロパティを設定する ◆ DataGrid などのバウンド コントロール用に [データ ソース] プロパティの ADO コントロールを指定する
ADO を使用したコードの記述	<ul style="list-style-type: none"> ◆ バウンド Data コントロールを使用して接続プロパティを設定する ◆ Data コントロールの Sub プロシージャの一部として SQL ステートメントを実行する

Pervasive PSQL を Delphi で使用

4

この章では、Delphi を使用して Pervasive PSQL データを表示および操作する方法について説明します。これには、ActiveX インターフェイスの Delphi IDE へのインストール、インストールされたインターフェイスの異なる機能の操作、Btrieve API を使用した基本的なファイル操作の実行、ODBC を使用した SQL クエリの実行に関する情報が含まれます。

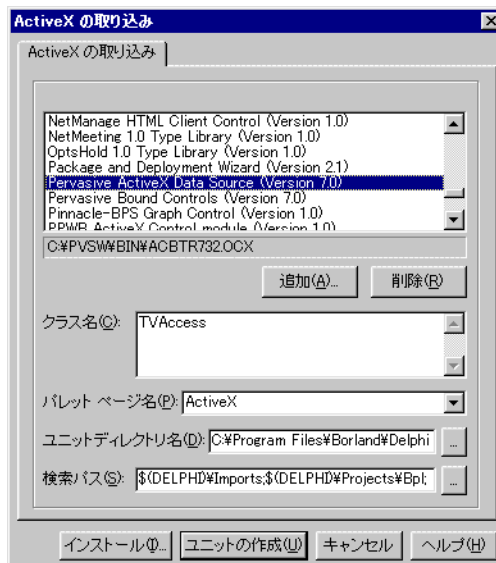
レッスン 1 : Delphi で ActiveX を使用する

このセクションでは、ActiveX インターフェイスをインストールして Delphi で操作する方法について説明します。作業項目は以下のとおりです。

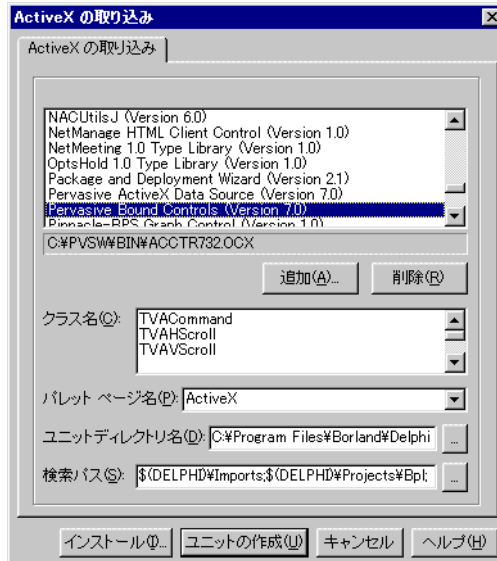
- 「Delphi IDE に ActiveX インターフェイスをインストールする」
- 「Delphi で VAccess コントロールを使用する」
- 「VAList ボックス コントロールを使用する」
- 「VAText コントロールを使用する」
- 「2つの VAccess コントロールを結合する」

Delphi IDE に ActiveX インターフェイスをインストールする

- 1 Delphi のオンライン ヘルプまたはマニュアルの指示に従い、Delphi のメイン メニューから [コンポーネント] をクリックして [ActiveX コントロールの取り込み] を選択します。
- 2 [ActiveX の取り込み] ダイアログ ボックスの一覧から、"Pervasive ActiveX Data Source (Version 1.0) " をクリックします。



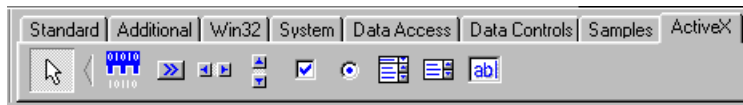
- 3 [インストール] をクリックし、指示に従ってコントロールとそのメンバー用の Delphi のラッパーを生成します。
- 4 このインポートの終了後、"Pervasive Bound controls (Version 1.0) " に対して同じ操作を繰り返します。



- 5 データ ソース コントロールとバウンド コントロールが、Delphi コンポーネント パレットの [ActiveX] ページに表示されていることを確認します。

Delphi で VAccess コントロールを使用する

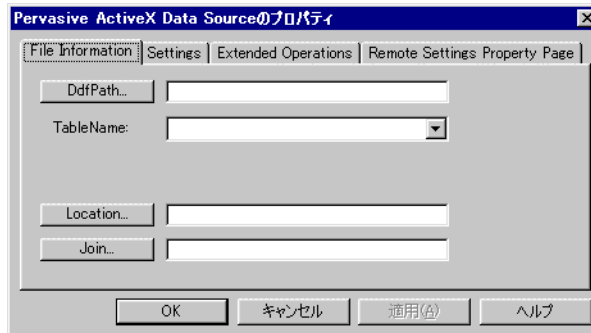
- 1 このコントロールをフォームまたはデータ モジュールに追加します。
 - a. Delphi IDE で、ActiveX インターフェイスを含むコンポーネントパレットのタブをクリックします。デフォルトでこのタブは、[ActiveX] という名前になっています。ActiveX のアイコンがパレットに表示されます。



- b. コントロールを選択してクリックします。
 - c. フォーム上で、コントロールを挿入する場所をクリック、またはマウスをドラッグしてコントロールのサイズと形を指定します。

2 このコントロールの **Btrieve File** プロパティを設定します。

- a. VAccess コントロールのプロパティ ダイアログ ボックスを開きます。Delphi 3 IDE の場合は、ActiveX コントロールを使用しているフォームを表示させます。コントロールを右クリックしてショートカット メニューから [プロパティ] を選択、またはそのコントロールをダブルクリックします。

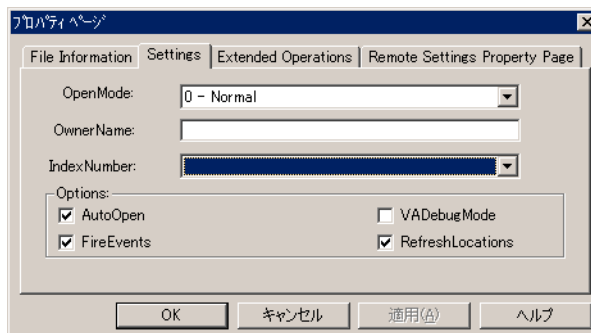


[File Information] タブで、以下のプロパティを設定します。

- **DdfPath** - コントロールに付属するデータ ファイルの定義を含む DDF の場所 (ドライブおよびパス)。このプロパティのデフォルト値は、VCBTRV.INI で設定されています。
- **TableName** - VAccess コントロールに付属するデータ ファイルの (DdfPath プロパティで指定した DDF 内の) テーブル名。
- **Location** - コントロールに付属するデータ ファイルのオペレーティング システム ファイル名。TableName プロパティで指定されたテーブルの場所がデフォルトになります。このプロパティには、完全修飾パスおよびファイル名、またはファイル名のみが入ります。パスを指定しなかった場合、コントロールは、DdfPath プロパティで指定された場所にあるデータ ファイルを開きます。

3 [Settings] タブをクリックして [IndexNumber] の値を確認します。

4 ボックスから、Btrieve ファイルのインデックスを選択します。



- 5 コードを追加してコントロールを初期化（クリア）します。Delphi のランタイム環境では、フォーム作成中に VAccess コントロールにアクセスしていない限り、バウンド コントロールは正しく初期化されません。
- Delphi IDE で、VAccess コントロールを使用しているフォームを選択します。
 - フォームのバックグラウンドをダブルクリックして FormCreate メソッドのコードを編集します。
 - エディターで、次のコードの VAccess1 の部分を自分のコントロールの名前に置き換え、プロシージャの begin ステートメントと end ステートメントの間に追加します。

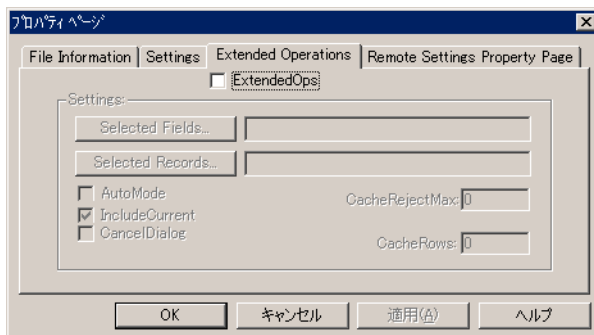
```
VAccess1.Clear;
```



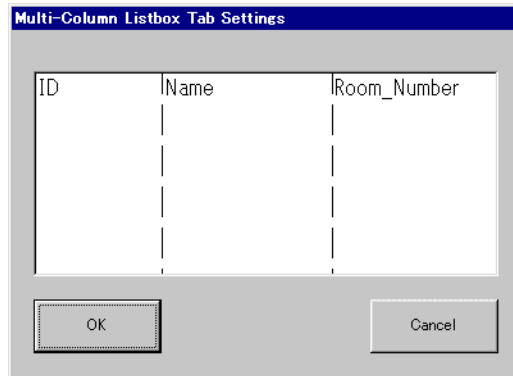
メモ 複数の VAccess コントロールを使用している場合は、各 VAccess コンポーネントに対して次のようなステートメントを begin ステートメントと end ステートメントの間に追加する必要があります。

```
VAccess2.Clear;
VAccess1.Clear;
```

- 6 Extended オペレーションを使用するかどうかを選択します。
- VAccess コントロールのプロパティダイアログボックスを開きます。
 - [Extended Operations] タブをクリックします。Extended オペレーションを開く場合は、[ExtendedOps] チェックボックスをオンにし、使用しない場合は、このチェックボックスをオフにします。



- 4 列幅を設定します。
 - a. [ColumnWidth]ボックスの横にある[...]ボタンをクリックします。
 - b. 編集ボックス内で、VAlst ボックスの最初の列の任意の右端をダブルクリックします。最後の列以外のすべての列に対してこの操作を繰り返します。この操作の終了後は、列の線を新しい位置までドラッグできます。



- 5 VAAutoScroll プロパティを設定します。

このプロパティを **False** に設定することにより、リストがオートフィルレコード リストとして使用されている場合 (**VARecordList = True**)、リスト内で現在選択されている項目が、自動的にそのリストの一番上にスクロールされなくなります。

レコード リスト VAlst ボックスのデフォルト (**VARecordList = True**) の動作では、リスト ボックスやその他の方法 (スクロール バーや **GetEqual** コマンドなど) によって位置が変更された場合でも、現在のレコードが必ずリストの一番上に表示されます。

VAAutoScroll が **False** の場合、このデフォルトは無効になり、リスト ボックス自体以外のソースによって位置が変更された場合にのみ、現在のレコードがリストの一番上に表示されます。たとえば、リスト ボックス内の項目をクリックすると **VAccess** コントロールの位置が変更されますが、リスト ボックス内の項目の位置は変更されません。

- 6 VARecordList プロパティを設定します。

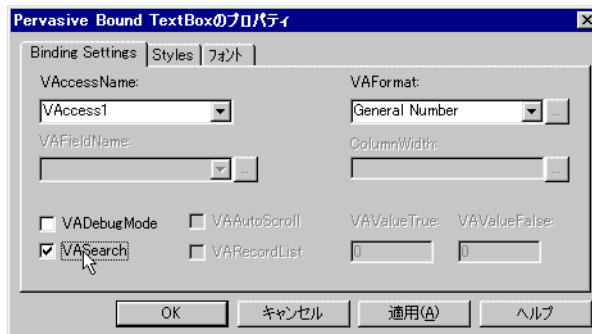
このプロパティを **True** に設定した場合は、リスト ボックス コントロールまたはコンボ ボックス コントロールのモードが、フィールド連結コントロールからレコード ブラウザーに切り替わります。

VARecordList プロパティにより、できるだけ多くのレコードをリストコントロールに入力することが可能になります。リスト内の一番上の項目は、現在のレコードです（ただし、VAAutoScroll プロパティが False に設定されている場合は、選択されているリスト項目が現在のレコード）。

VARecordList = True の場合は、追加機能を使用することができ、この追加機能には、設計時にプロパティ ページのボタンを使用してアクセスできます。VAFieldName プロパティおよび VAFormat プロパティには、それぞれセミコロンで区切られたフィールドと書式のリストを含めることができます。ただし、コントロールがフィールド バウンドモードの場合（VARecordList プロパティが False の場合）は、1つのエントリに制限されます。また、VAAutoScroll プロパティと ColumnWidth プロパティが使用可能になります。

VAText コントロールを使用する

- 1 コントロールをフォームに追加します。「このコントロールをフォームまたはデータ モジュールに追加します。」を参照してください。
- 2 「バウンド コントロールの VAccess プロパティを設定します。」を参照してください。
 - a. バウンド コントロールのプロパティ ダイアログ ボックスを開きます。
 - b. [VAccessName] ボックスの矢印をクリックします。
 - c. ドロップダウン リストからコントロールを選択します。

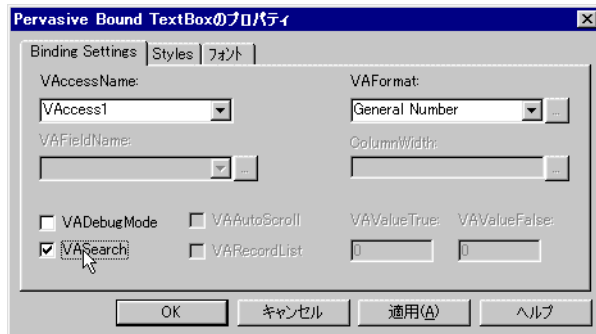


- 3 VAFieldName プロパティを設定します。
 - a. バウンド コントロールのプロパティ ダイアログボックスを開きます。
 - b. [VAFieldName] ボックスの矢印をクリックします。
 - c. ドロップダウン リストからフィールドを選択します。

4 VASearch プロパティを設定します。

このプロパティを True に設定した場合、VAccess コントロールは、データソースの IndexNumber プロパティによって指定されたインデックスの最初のセグメントのテキストボックスの値を使用し、そのテキストが変化することによって GetGreaterOrEqual メソッドを実行します。VAFieldName プロパティの設定は、無効になります。

- a. バウンド コントロールのプロパティ ダイアログ ボックスを開きます。
- b. このプロパティを True に設定する場合は、[VASearch] チェックボックスをオンにします。
- c. False に設定する場合は、このチェックボックスをオフにします。



2 つの VAccess コントロールを結合する

VAccess コントロールの Join プロパティにより、キー値を使用して 2 つのファイルをリンクさせることが可能になり、たとえば、最初の (マスター) ファイルでのレコードの位置を変更した場合、その変更は 2 番目の (スレーブ) ファイルから取得されたレコードにも反映されます。Join プロパティは、リレーショナル データベースを参照するための簡単な機能を提供します。

スレーブ コントロール内の結合されたフィールドは、インデックス フィールドである必要があります。またこのフィールドのデータは、マスター コントロール内の対応するフィールドと同じ未処理データである必要があります。マスター コントロール内のフィールドは、インデックスである必要はありません。スレーブ コントロールの IndexNumber プロパティは、この結合に関連があるインデックスに設定されたままであることが条件で、設定されていない場合、この結合は解除されます。

1 対 1、多対 1、1 対多のすべての結合が使用できます。スレーブ VAccess コントロールの ExtendedOps プロパティが True に設定されている場合は、1 対多の関係が最も有効です。この場合、スレーブ コントロールは、キー値に一致するすべてのレコードをマスター コントロールから自動的に取得し、その RowColumnValue プロパティからフィールド データにアクセス可能になります。このデータは、VAccessName プロパティをスレーブ コ

トロールの名前に、また **VAFldName** プロパティを表示されるフィールドにそれぞれ設定することにより、レコード リスト モードで **VAList** ボックスに表示させることができます。

- 1 スレーブ コントロールの **Join** プロパティを設定します。
- 2 スレーブ コントロールのプロパティ ダイアログ ボックスを開きます。

Delphi 3 IDE では、**ActiveX** コントロールを使用しているフォームを表示させ、そのコントロールを右クリックしてショートカット メニューから [**プロパティ**] を選択、またはそのコントロールをダブルクリックします。

- 3 [**File Information**] タブの [**Join**] をクリックします。
- 4 [**To VAccess**] ボックスの矢印をクリックします。
- 5 ドロップダウン リストからマスター ファイルを選択します。
- 6 [**Where**] ボックスで、マスター ファイルをリンクさせるスレーブ ファイルのインデックスを選択します。
- 7 [**Fields**] ボックスで、スレーブ ファイルのリンク先となるマスター ファイルのフィールドをダブルクリックします。
- 8 [**OK**] をクリックして [**Join Controls**] ダイアログ ボックスを閉じます。

レッスン 2 : Delphi で Btrieve API を使用する

このセクションでは、Delphi での Btrieve API の使用について説明します。作業項目は以下のとおりです。

- 基本的なファイル操作を実行する
 - ファイルを開く
 - ファイルを閉じる
 - ファイルの作成
- データの変更
 - レコードの挿入
 - レコードの更新
 - レコードの削除
- データの取得
 - Step オペレーションの実行
 - Get オペレーションの実行
 - セグメント化されたインデックスの処理
 - データを検索する

基本的なファイル操作を実行する

ファイルを開く

Btrieve ファイルを開くには、Open (0) オペレーションおよび以下の情報を指定して BTRV を呼び出します。

- a. ファイルのポジション ブロック
- b. ファイル名
- c. ファイルのオーナー ネーム

BTRV 呼び出しをするために、ファイル名をキー バッファに入力し、オーナー ネームをデータ バッファに入力します。両方のパラメーターは、「ヌルで終了」している必要があります。DataLength パラメーターには、データ バッファの長さを入力する必要があります。

ポジション ブロックは、ファイルが開いている間、有効なスコープで宣言する必要があります。ポジション ブロックは、関数の呼び出し中のカーンシー データおよびその他の情報を保存するために使用されるため、この変数は、ファイルに関連するすべての Btrieve 呼び出しから有効かつ参照可能である必要があります。

例

```
var
    Status:smallint;
    PosBlock:array[1..128] of byte;
    DataBuffer:array[0..8] of char;
    KeyBuffer:array[0..254] of char;
    DataLength:word;
    KeyNumber:byte;
    [ . . . ]
// オーナー ネーム (ある場合) を DataBuffer に設定し、長さも
// 指定する
fillchar(DataBuffer, sizeof(DataBuffer), #0);
DataBuffer := ''; // このファイルには、オーナー ネームなし
DataLength := length(DataBuffer);
// KeyBuffer にファイル名を入力する
fillchar(KeyBuffer, sizeof(KeyBuffer), #0);
KeyBuffer := 'g:¥psql¥sdk¥pviddb¥data¥customer.mkd';
// KeyNumber を 0 に設定
KeyNumber := 0;
// パラメーターを指定して Btrieve を呼び出す
Status := BTRV( B_OPEN,
                PosBlock,
                DataBuffer,
                DataLength,
                KeyBuffer,
                KeyNumber);

// 返されたステータスを確認
ShowMessage('Open returned ' + intToStr(Status));
[ . . . ]
```

ファイルを閉じる

Btrieve ファイルを閉じるには、Close (1) オペレーションで BTRV を呼び出し、開いているファイルのポジションブロックを渡します。ステータスが返され、この操作が正しく完了したかどうかを確認できます。

例

```
[ . . . ]
Status := BTRV( B_CLOSE,
                PosBlock,
                DataBuffer,
                DataLength,
```

```

        KeyBuffer,
        KeyNumber);
ShowMessage('Close returned ' + intToStr(Status));
[. . .]

```

ファイルを作成する

ファイルを作成するには、新しい **Btrieve** ファイルの作成に必要な情報を含む構造体を作成する必要があります。この構造体を作成する最も簡単な方法は、**BTRV** 呼び出しが実行される前に集計される複数の小さい構造体を作成することです。**Create** (14) 関数呼び出しに渡される構造体についての詳細は、プログラマ用のドキュメントを参照してください。

例

```

{*****
  Stat および Create 処理用のレコードの型定義
*****}
type
// ファイル仕様 - ファイル 1 つにつき 1 つの構造体
FILE_SPECS = packed record
  recLength      :smallint;
  pageSize      :smallint;
  indexCount    :smallint;
  reserved      :array[0..3] of char;
  flags         :smallint;
  dupPointers   :byte;
  notUsed       :byte;
  allocations   :smallint;
end;

// キー仕様 - ファイル内のキー セグメント 1 つにつき 1 つの構造体
KEY_SPECS = packed record
  position      :smallint;
  length        :smallint;
  flags         :smallint;
  reserved      :array[0..3] of char;
  keyType       :char;
  nullChar      :char;
  notUsed       :array[0..1] of char;
  manualKeyNumber :byte;
  acsNumber     :byte;
end;

```

Pervasive PSQL を Delphi で使用

```
// 集合バッファーには、1 つの FILE_SPECS と、
// キー セグメントと同数の KEY_SPECS 構造体が入る
FILE_CREATE_BUF = packed record
    fileSpecs :FILE_SPECS;
    // ここでは、拡張性を考慮して配列を使用
    keySpecs  :array[1..1] of KEY_SPECS; // 1 つのキー セグメント
end;
{*****}
    ファイル仕様構造体を作成して値を入力
{*****}
var
    NewFileSpec :FILE_CREATE_BUF;
    DataLength  :smallint;
    KeyBuffer   :array[0..254] of char;
    KeyNumber   :byte;
    PosBlock    :array[1..128] of byte;
[. . .]
fillchar(NewFileSpec, sizeof(NewfileSpec), #0);
With NewFileSpec.fileSpecs do begin
    recLength   := 100;
    pageSize    := 4096;
    indexCount  := 3; // インデックスの数 (キー セグメント数ではない)
    reserved    := 0;
    flags       := 0;
    dupPointers := 0;
    notUsed     := 0;
    allocations := 0;
end;

With NewFileSpec.keySpecs[1] do begin
    position    := 0;
    length      := 4;
    flags       := 0;
    // 予約済み
    // 必要なし - fillchar() によって NULL 化済み
    keyType     := 1;
    nullChar    := 0;
    // 使用せず
    // 必要なし - fillchar() によって NULL 化済み
    manualKeyNumber := 0;
    acsNumber   := 0;
```

```

end;
{*****}
    ファイル パスとファイル名の指定
*****}
fillchar(KeyBuffer, sizeof(KeyBuffer), #0);
KeyBuffer := 'c:\ttest.mkd';
{*****}
    Key Number を設定。警告せずに上書きする場合は 0、ファイルが
    存在する場合にエラーとする場合は -1。
*****}
KeyNumber := 0;
{*****}
    Btrieve Create オペレーションを呼び出し、返されたステータス
    を確認
*****}
Status := BTRV( B_CREATE,
                PosBlock,
                NewFileSpec,
                DataLength,
                KeyBuffer,
                KeyNumber);
ShowMessage('Create returned ' + intToStr(Status));

```

データの変更

レコードの挿入

- 1 挿入する Btrieve レコードをデータ バッファに配置します。
- 2 Btrieve Insert 関数を呼び出し、返されたステータスを確認します。

例

```

Status := BTRV( B_INSERT,
                PosBlock,
                DataBuffer,
                DataLength,
                KeyBuffer,
                KeyNumber);
ShowMessage('Insert returned ' + intToStr(Status));

```

レコードの更新

- 1 Get オペレーションまたは Step オペレーションを使用してカレンシーを確立し、レコードを読み込みます。
- 2 データ バッファーを変更します。
- 3 Btrieve Update 関数を呼び出します。

例

```
Status := BTRV( B_UPDATE,
                PosBlock,
                DataBuffer,
                DataLength,
                KeyBuffer,
                KeyNumber);
ShowMessage('Update returned ' + intToStr(Status));
```

レコードの削除

- 1 Get オペレーションまたは Step オペレーションを使用してカレンシーを確立し、レコードを読み込みます。
- 2 Btrieve Delete 関数を呼び出します。

例

```
Status := BTRV( B_DELETE,
                PosBlock,
                DataBuffer,
                DataLength,
                KeyBuffer,
                KeyNumber);
ShowMessage('Delete returned ' + intToStr(Status));
```

データの取得

Step オペレーションを実行する

Step オペレーションは、インデックスを使用せずにレコードを返します。

- 1 Step オペレーションを呼び出します。

```
Status := BTRV( B_STEP_FIRST,
                PosBlock,
                DataBuffer,
                DataLength,
                KeyBuffer,
                KeyNumber);
```

2 ステータスを確認します。

```
ShowMessage('Update returned ' + intToStr(Status));
```

3 データ バッファを使用または操作します。

Get オペレーションを実行する

Get オペレーションは、インデックスを使用してレコードを返します。

1 (Get Equal、Get Greater、Equal 用などに) キー バッファのパラメータを適切な値に設定します。

```
// 次のコードは、2 つのセグメント (両方とも 10 文字) を持つ
// インデックスのキー バッファを設定します。
// ZStrings とは違い、これらのフィールドはヌル終端でなく、
// 空白を詰めます。
```

```
KeyBuffer := 'Adams      George    ';
```

2 特定のインデックスにキー番号パラメータを設定します。

```
KeyNumber := 1;
```

3 Step オペレーションを呼び出します。

```
Status := BTRV( B_GET_EQUAL,
                PosBlock,
                DataBuffer,
                DataLength,
                KeyBuffer,
                KeyNumber);
```

4 ステータスを確認します。

```
ShowMessage('Update returned ' + intToStr(Status));
```

5 データ バッファを使用または操作します。

セグメント インデックスを使用する

セグメント インデックスは、セグメントを持たないインデックスと同じように使用できますが、キーバッファに値を入力する際に注意が必要です。

キー バッファには、キーと同じ形式で値を入力する必要があります。整数フィールドおよびその他のバイナリ フィールドでは、キー バッファの情報は、バイナリ形式である必要があります。それ以外の文字列型では、データは、**Btrieve** ファイルに表示される形式とまったく同じ形式で表示される必要があります。

キー バッファのレコードは、**Variant Record** (「C」では、「共用体」として定義すると便利です。次のキー バッファの種類は、異なる複数のセグメント キー タイプおよびセグメント化されていないキー タイプに対して使用できます。

例

```

type
  KEY_BUFFER_TYPE = packed record
    case integer of 1: (
      IDNumber:integer;
      Filler1:array[1..255 - sizeof(integer)] of char;
    );
    2: (
      LastName:array[0..23] of char;
      FirstName:array[0..23] of char;
      Filler2:array[1..255 - 24 - 24] of char;
    );
    3: (
      ZipCode:array[0..9] of char;
      Filler3:array[1..255 - 10] of char
    )
  end;
[. . .]
var
  KeyBuffer:KEY_BUFFER_TYPE;
[. . .]
fillchar(KeyBuffer, sizeof(KeyNumber), #0);
KeyBuffer.IDNumber := 1047;
Status := BTRV( B_GET_EQUAL,
               PosBlock,
               DataBuffer,
               DataLength,
               KeyBuffer,
               KeyNumber);

```

データを検索する

データを検索するには、キーを使用してデータベース内の行を検出します。実際の行ではなくファイル内のキーのみを検出(存在しているものと仮定)するには、GetEqual、GetNext、GetFirst、GetLast のいずれかのオペレーション番号に +50 (btrconst.pas の KEY_BIAS) を追加します。

例

```

// キーのみを取得 -- レコードは取得しない
Status := BTRV( B_GET_EQUAL + KEY_BIAS,
               PosBlock,
               DataBuffer,
               DataLength,
               KeyBuffer,
               KeyNumber);

```

レッスン 3 : Delphi で ODBC を使用する

データベースに接続する

- 1 TDatabase (オプション) と、TTable または TQuery、および TDataSource をフォーム上に配置します。

TDatabase オブジェクトをフォーム上に配置してプロパティを設定するには - Delphi

- a. コンポーネントをフォーム上に配置します。
- b. AliasName プロパティを ODBC DSN に設定します。
- c. DatabaseName プロパティを覚えやすい値に設定します。
- d. LoginPrompt プロパティを適切な値に設定します。

TQuery をフォーム上に配置してプロパティを設定するには

- a. コンポーネントをフォーム上に配置します。
 - b. DatabaseName プロパティを TDatabase に指定した名前に設定します。
 - c. SQL プロパティを有効な SQL ステートメント (パラメーターの有無は無関係) に設定します。
 - d. SQL でパラメーターが使用されている場合は、Params 文字列を設定します。
 - e. 設計時にクエリを実行できるように、Active プロパティを True に設定します。
- 2 TDatabase のプロパティで Connected を True に設定します。
 - 3 TTable または TQuery のプロパティで、Active を True に設定します。
 - 4 TTable または TQuery へ接続するように TDataSource を設定します。
 - 5 プログラムの起動時に自動接続されないようにするには、TTable または TQuery の Active プロパティを False に設定しておき、実行時に接続が要求された際、コードでこの設定を True に変更します。

SQL を使用してクエリを実行する

- 1 TQuery コンポーネントをオプションの TDatabase と TDataSource と一緒に使用し、TQuery の SQL プロパティをクエリを表す有効な SQL ステートメントに設定します。
- 2 TQuery の Active プロパティを True に設定して、設計時にクエリを実行したり、実行時に TQuery を閉じたり開いたりできるようにします。

例

```
begin
  TitleQuery.Active := FALSE;
  TitleQuery.SQL.Clear;
  if IDRadioButton.Checked then begin
    TitleQuery.SQL.Add('select a.*, b.Description from
      Titles a, Categories b ');
    TitleQuery.SQL.Add('where CategoryID = Category ');
    TitleQuery.SQL.Add('order by a.TitleID');
  end else begin
    TitleQuery.SQL.Add('select a.*, b.Description from
      Titles a, Categories b ');
    TitleQuery.SQL.Add('where CategoryID = Category ');
    TitleQuery.SQL.Add('order by a.Title');
  end;
  TitleQuery.Active := TRUE;
end;
```

コントロールをバインドする

```
Query1.Params[1] := 'Clyde';
Query1.Params[2] := '1234 First Street';
Query1.Params[3] := '';
Query1.Params[4] := 'Austin';
Query1.Params[5] := 'TX';
Query1.Params[6] := '78743';
Query1.Params[7] := '512-555-1234';
Query1.ExecSQL;
```

SQL を使用して更新する

- 1 TQuery コンポーネントを使用します。
- 2 SQL プロパティに Update ステートメントを設定します。
- 3 ExecSQL メソッドを呼び出します。

例

```
TitleUpdateQuery.SQL.Clear;
TitleUpdateQuery.SQL.Add('UPDATE titles ');
TitleUpdateQuery.SQL.Add('set TitleID = ' +
  intToStr(Titles.TitleQuery.FieldValues['TitleID'])
  + ', ');
```

```

TitleUpdateQuery.SQL.Add('Title = ''' + TitleEdit.Text
    + ''', ');
TitleUpdateQuery.SQL.Add('Category = ' +
    intToStr(CategoryDBComboBox.ItemIndex + 1) + ', ');
TitleUpdateQuery.SQL.Add('Price = ' + PriceEdit.Text +
    ', ');
TitleUpdateQuery.SQL.Add('Term = ' + TermEdit.Text +
    ');
TitleUpdateQuery.SQL.Add('where TitleID = ' +
    intToStr(Titles.TitleQuery.FieldValues['TitleID'])
    + ' ');
TitleUpdateQuery.ExecSQL;

```

SQL を使用して挿入する

- 1 TQuery コンポーネントをフォーム上に配置します。
- 2 SQL プロパティに Insert ステートメントを設定します。
- 3 TQuery の ExecSQL メソッドを呼び出します。

例

```

TitleUpdateQuery.SQL.Clear;
TitleUpdateQuery.SQL.Add('INSERT into titles values (');
TitleUpdateQuery.SQL.Add('0, ''' + TitleEdit.Text + ''',
    ');
TitleUpdateQuery.SQL.Add(intToStr(CategoryDBComboBox.It
    emIndex + 1) + ', ');
TitleUpdateQuery.SQL.Add(PriceEdit.Text + ', ');
TitleUpdateQuery.SQL.Add(TermEdit.Text + ' ')');
TitleUpdateQuery.ExecSQL;

```

パラメーター クエリを使用する

- 1 TQuery コンポーネントを使用してください。
- 2 SQL プロパティにパラメーターを持つステートメントを設定します。
パラメーターは、コロンで始まる :Param で表されます。
- 3 Params プロパティ エディターで、SQL ステートメント内の名前付け
された各パラメーターの Type とデフォルト値を指定します。
- 4 実行時に、Params プロパティをコードで設定します。

例

```
Query1.Close;
Query1.SQL.Add('select * from customers ');
Query1.SQL.Add('where LastName = :LName and FirstName =
    :FName');
[ . . . ]
Query1.Params[0] := 'Henderson';
Query1.Params[1] := 'Clyde';
Query1.Open;
```

パラメーター Update ステートメントを使用する

- 1 TQuery コンポーネントを使用してください。
- 2 SQL プロパティにパラメーターを持つ Update ステートメントを設定します。パラメーターは、コロンで始まる :Param で表されます。
- 3 Params プロパティ エディターで、SQL ステートメント内の名前付けされた各パラメーターの Type とデフォルト値を指定します。
- 4 実行時に、Params プロパティをコードで設定します。

例

```
Query1.Close;
Query1.SQL.Clear;
Query1.SQL.Add('update customers ');
Query1.SQL.Add('set LastName = :LName, FirstName =
    :FName ');
Query1.SQL.Add('where CustID = :CustID');
[ . . . ]
Query1.Params[0] := 'Henderson';
Query1.Params[1] := 'Clyde';
Query1.Params[2] := iCustID;
Query1.ExecSQL;
```

パラメーター Insert ステートメントを使用する

- 1 TQuery コンポーネントをフォーム上に配置します。
- 2 SQL プロパティにパラメーターを持つ Insert ステートメントを設定します。パラメーターは、コロンで始まる :Param で表されます。
- 3 Params プロパティ エディターで、SQL ステートメント内の名前付けされた各パラメーターの Type とデフォルト値を指定します。
- 4 実行時に、Params プロパティをコードで設定します。

例

```
Query1.Close;
Query1.SQL.Clear;
Query1.SQL.Add('insert into customers ');
Query1.SQL.Add('values (0, :LName, :FName, :Address1,
    :Address2, ');
Query1.SQL.Add(':City, :State, :Zip, :Phone));
[ . . . ]
Query1.Params[0] := 'Henderson';
```


Pervasive PSQL を Visual Basic で使用

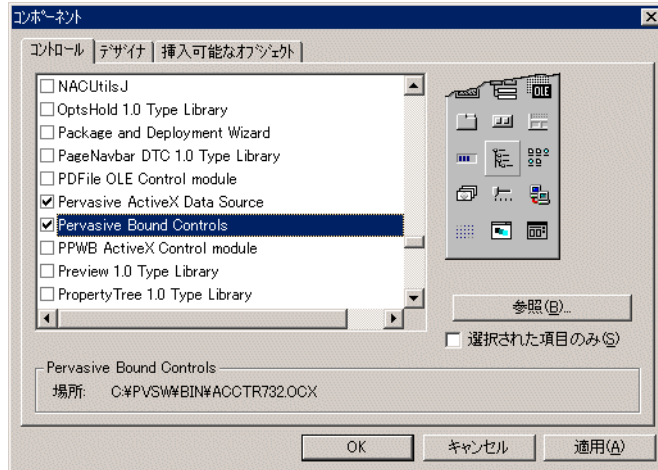
5

この章では、Microsoft Visual Basic を使用して Pervasive PSQL データを表示および操作する方法について説明します。この中には、ActiveX インターフェイスのインストールおよび操作、Btrieve API での基本的なファイル操作の実行、ODBC を使用した SQL クエリの実行に関する情報が含まれます。

レッスン 1 : Visual Basic で ActiveX を使用する

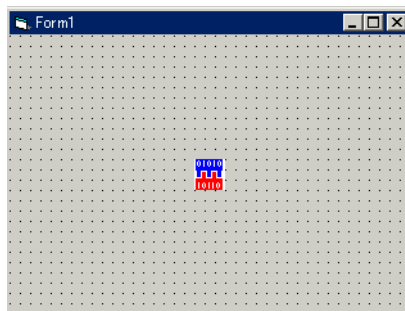
ActiveX インターフェイスを準備する

- 1 "Pervasive ActiveX Data Source" と "Pervasive Bound Controls" を Visual Basic のツールバーに追加します。



Visual Basic の VAccess コントロールを使用する

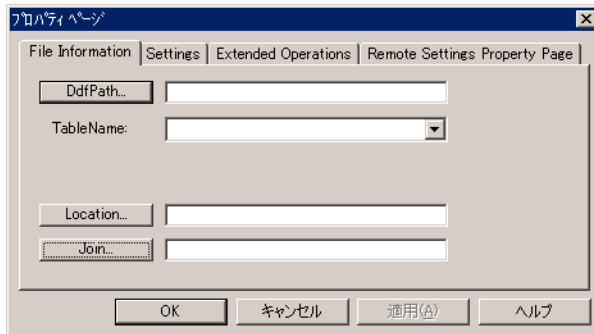
- 1 VAccess データ コントロールを選択してフォームに配置します。



- 2 このコントロールの Btrieve File プロパティを設定します。
 - a. コントロールを右クリックして [プロパティ] を選択します。または、[表示 | プロパティ ページ] を選択します。

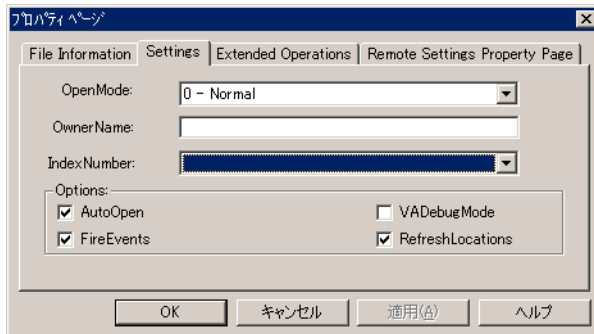
b. [File Information] タブで、以下のプロパティを設定します。

- **DdfPath** - コントロールに付属するデータ ファイルの定義を含む DDF の場所 (ドライブおよびパス)。
- **TableName** - VAccess コントロールに付属するデータ ファイルの (DdfPath プロパティで指定した DDF 内の) テーブル名。
- **Location** - コントロールに付属するデータ ファイルのオペレーティング システム ファイル名。TableName プロパティで指定されたテーブルの場所がデフォルト になります。このプロパティには、完全修飾パスおよびファイル名、またはファイル名のみが入ります。パスを指定しなかった場合、DdfPath プロパティで指定された場所にあるデータ ファイルが開きます。



3 [Settings] タブをクリックして [IndexNumber] の値を確認します。

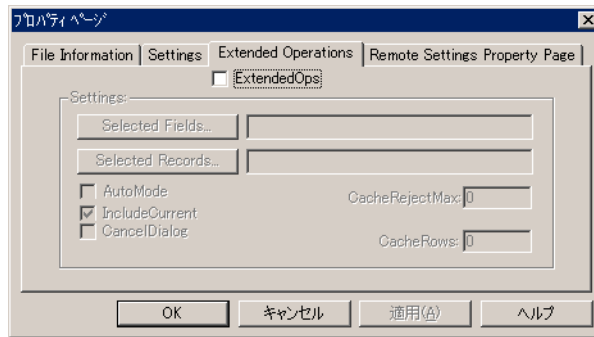
4 ボックスから、Btrieve ファイルのインデックスを選択します。



5 Extended オペレーションを使用するかどうかを選択します。

アプリケーションで Extended オペレーションを使用すると、Btrieve サーバーから一度に複数のレコードを取得することができ、フィールド値に基づいてサーバー側でフィルタリングを行うことができます。Extended オペレーションにより、レコード取得操作によってはパフォーマンスが非常に向上します。

[Extended Operations] タブをクリックします。Extended オペレーションを使用する場合は、[ExtendedOps] チェック ボックスをオンにし、使用しない場合は、このチェック ボックスをオフにします。



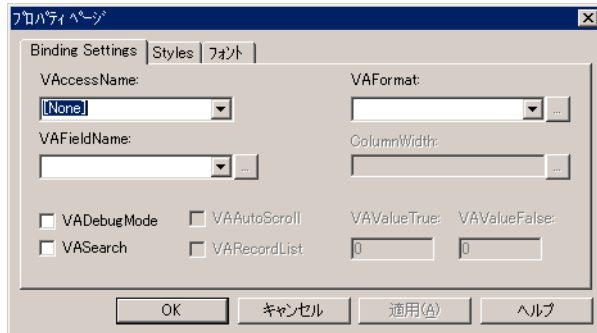
- 6 [OK] をクリックします。
- 7 次のコードで VAccess1 の部分を自分のコントロールの名前に置き換え、Form_Load Event に追加します。

```
VAccess1.GetFirst
```

必要な場合は、初期化クエリのコードを追加します。

VAText コントロールを使用する

- 1 フォームにコントロールを追加します。「[VAccess データ コントロールを選択してフォームに配置します。](#)」を参照してください。
- 2 バウンド コントロールの VAccess プロパティを設定します。「[バウンド コントロールの VAccess プロパティを設定します。](#)」を参照してください。
 - a. バウンド コントロールのプロパティ ダイアログ ボックスを開きます。
 - b. [VAccessName] ボックスの矢印をクリックします。
 - c. ドロップダウン リストからコントロールを選択します。
- 3 VAFieldName プロパティを設定します。
 - a. [VAFieldName] ボックスの矢印をクリックします。
 - b. ドロップダウン リストからフィールドを選択します。



4 VASearch プロパティを設定します。

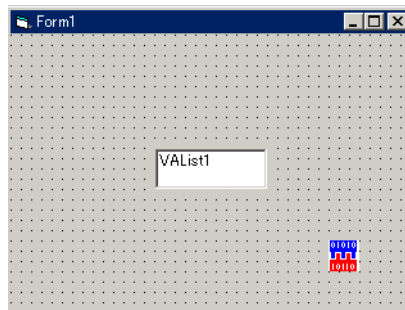
このプロパティを True に設定した場合、VAccess コントロールは、データソースの IndexNumber プロパティによって指定されたインデックスの最初のセグメントのテキスト ボックスの値を使用し、そのテキストが変化することによって GetGreaterOrEqual メソッドを実行します。VAFieldName プロパティの設定は、無効になります。

- a. このプロパティを True に設定する場合は、[VASearch] チェック ボックスをオンにします。
- b. False に設定する場合は、このチェック ボックスをオフにします。

5 [OK] をクリックします。

VAList コントロールを使用する

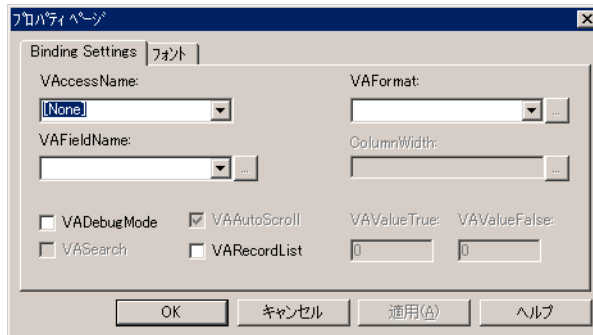
1 フォームにコントロールを追加します。



2 バウンド コントロールの VAccess プロパティを設定します。

- a. VAList コントロールのプロパティ ダイアログ ボックスを開きます。
- b. [VAccessName] ボックスの矢印をクリックします。

- c. ドロップダウン リストからデータ ソース コントロールを選択します。



- 3 VAAutoScroll プロパティを設定します。

このプロパティを **False** に設定することにより、リストがオートフィルレコード リストとして使用されている場合 (**VARecordList = True**)、リスト内で現在選択されている項目が、自動的にそのリストの一番上にスクロールされなくなります。

レコード リスト **VAList** のデフォルト (**VARecordList = True**) の動作では、リスト ボックスやその他の方法 (スクロール バーや **GetEqual** コマンドなど) によって位置が変更された場合でも、現在のレコードが必ずリストの一番上に表示されます。

VAAutoScroll が **False** の場合、このデフォルトは無効になり、リスト ボックス自体以外のソースによって位置が変更された場合にのみ、現在のレコードがリストの一番上に表示されます。たとえば、リスト ボックス内の項目をクリックすると **VAccess** コントロールの位置が変更されますが、リスト ボックス内の項目の位置は変更されません。

- 4 VARecordList プロパティを設定します。

このプロパティを **True** に設定した場合は、リスト ボックス コントロールまたはコンボ ボックス コントロールのモードが、フィールド連結コントロールからレコード ブラウザーに切り替わります。

VARecordList プロパティにより、できるだけ多くのレコードをリスト コントロールに入力することが可能になります。リスト内の一番上の項目は、現在のレコードです (ただし、**VAAutoScroll** プロパティが **False** に設定されている場合は、選択されているリスト項目が現在のレコード)。

VARecordList = True の場合は、追加機能を使用することができ、この追加機能には、設計時にプロパティ ページのボタンを使用してアクセスできます。**VAFieldName** プロパティおよび **VAFormat** プロパティには、それぞれセミコロンで区切られたフィールドと書式のリストを含

めることができます。ただし、コントロールがフィールド バウンド モードの場合 (VARecordList プロパティが False の場合) は、1 つのエントリに制限されます。また、VAAutoScroll プロパティと ColumnWidth プロパティが使用可能になります。

5 表示させるフィールドを選択します。

VAList に表示させるフィールドを 1 つ選択するには

- a. [VAFieldName] ボックスの矢印をクリックします。
- b. ドロップダウン リストからフィールドを選択します。

VAList に表示させるフィールドを複数選択するには

- a. [VAFieldName] ボックスのテキスト フィールドをクリックします。
- b. 表示させるフィールドをセミコロン (;) で区切って入力します。

または、

- a. [VAFieldName] テキスト ボックスの横にある [...] ボタンをクリックします。
- b. 追加するフィールドをそれぞれダブルクリックします。
- c. [OK] をクリックします。

6 列幅を設定します。

- a. [ColumnWidth] ボックスの横にある [...] ボタンをクリックします。
- b. 編集ボックス内で、VAList ボックスの最初の列の任意の右端をダブルクリックします。最後の列以外のすべての列に対してこの操作を繰り返します。この操作の終了後は、列の線を新しい位置までドラッグできます。

2 つの VAccess コントロールを結合する

VAccess コントロールの Join プロパティを使用すると、キー値を使用して 2 つのファイルをリンクさせることができ、たとえば、最初の (マスター) ファイルでのレコードの位置を変更した場合、その変更は 2 番目の (スレーブ) ファイルから取得されたレコードにも反映されます。Join プロパティは、リレーショナル データベースを参照するための簡単な機能を提供します。

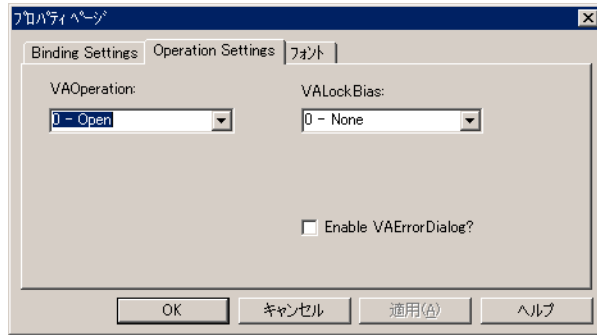
スレーブ コントロール内の結合されたフィールドは、インデックス フィールドである必要があります。またこのフィールドのデータは、マスター コントロール内の対応するフィールドと同じ未処理データである必要があります。マスター コントロール内のフィールドは、インデックスである必要はありません。スレーブ コントロールの IndexNumber プロパティは、この結合に関連があるインデックスに設定されたままであることが条件で、設定されていない場合、この結合は解除されます。

1 対 1、多対 1、1 対多のすべての結合が使用できます。スレーブ VAccess コントロールの ExtendedOps プロパティが True に設定されている場合は、1 対多の関係が最も有効です。この場合、スレーブ コントロールは、キー値に一致するすべてのレコードをマスター コントロールから自動的に取得し、その RowColumnValue プロパティからフィールド データにアクセス可能になります。このデータは、VAccessName プロパティをスレーブ コントロールの名前に、また VAFieldName プロパティを表示されるフィールドにそれぞれ設定することにより、レコード リスト モードで VAList ボックスに表示させることができます。スレーブ コントロールの Join プロパティを設定するには、以下の操作を行います。

- 1 スレーブ コントロールのプロパティ ダイアログ ボックスを開きます。
- 2 [Setting] タブをクリックします。
- 3 インデックスを設定します。
- 4 [File Information] タブをクリックします。
- 5 [File Information] タブの [Join] をクリックします。
- 6 [To VAccess] ボックスの矢印をクリックします。
- 7 ドロップダウン リストからマスター ファイルを選択します。
- 8 [Where] ボックスで、マスター ファイルをリンクさせるスレーブ ファイルのインデックスを選択します。
- 9 [Fields] ボックスで、スレーブ ファイルのリンク先になるマスター ファイルのフィールドをダブルクリックします。
- 10 [OK] をクリックして [Join Controls] ダイアログ ボックスを閉じます。
- 11 [OK] をクリックして VAccess のプロパティ ページを閉じます。

VACommand コントロールを使用して Btrieve オペレーションを実行する

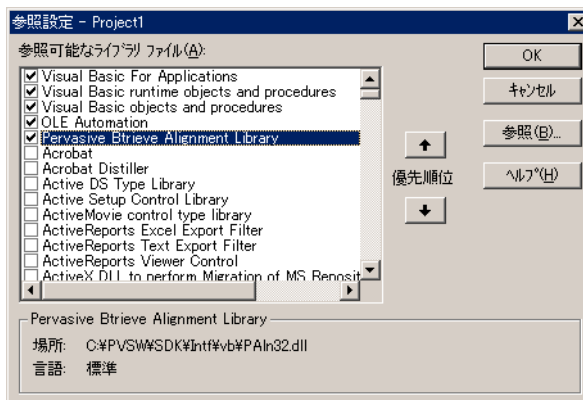
- 1 フォームにコントロールを追加します。「[VAccess データ コントロールを選択してフォームに配置します。](#)」を参照してください。
- 2 VAccess プロパティを設定します。「[バウンド コントロールの VAccess プロパティを設定します。](#)」を参照してください。
- 3 VAOperation プロパティを設定します。
 - a. プロパティ ダイアログ ボックスの [Operation Settings] タブをクリックします。
 - b. [VAOperation] ボックスの矢印をクリックします。
 - c. ドロップダウン リストからオペレーションを選択します。



レッスン 2 : Visual Basic で Btrieve API を使用する

Visual Basic プロジェクトで PALN32.DLL を参照させる

- 1 [プロジェクト] メニューの [参照設定] をクリックします。
- 2 [Pervasive Btrieve Alignment Library] のモジュールを選択します。このモジュールが表示されていない場合は、[参照] ボタンをクリックしてこのモジュールをリストに追加します。このモジュールは、Btrieve API 用の Visual Basic サンプル アプリケーションと同じディレクトリにインストールされます。



- 3 %Program Files%\Pervasive Software\PSQL\SDK\INT\FVB (デフォルトの場所にダウンロードした場合) から BTR32VB.BAS をプロジェクト ディレクトリにコピーし、プロジェクトに追加します。

グローバル構造体を初期化する

Pervasive Btrieve Alignment ライブラリ (PALN32.DLL) は、Pervasive PSQL SDK に用意されています。このライブラリは、アライメント調整された構造体のパックや、データベースの行のアンパックに使用されます。

データベースをアクセス可能にするには、データ操作のためのデータ構造体が必要です。以下の項目を保存する必要があります。

- データベース形式のパックされたデータ
- Visual Basic の UDT 形式のアンパック データ
- パックおよびアンパックに必要なフィールドの情報

- 1 パックされたデータベースの行を保存するために必要な構造体を作成します。

```

Const CustRowSize = 193
Type CustRowType
    buf(1 To CustRowSize) As Byte
End Type
Dim CustRow As CustRowType

```

- 2 保持しているデータの構造体を作成します。

```

Public Type CustRecType
    custid As Long
    lastname As String * 24
    firstname As String * 24
    address1 As String * 36
    address2 As String * 36
    city As String * 24
    state As String * 2
    zip As String * 10
    homephone As String * 12
    workphone As String * 12
    status As Long
    member As Byte
    expiration As Long
End Type

```

- 3 保持しているファイル構造の構造体を作成します。

```
Global CustFldMap(1 To 13) As FieldMap
```

- 4 ファイル構造をメモリに読み込みます。

```

'Customer フィールド マップを初期化する
DDF ファイルが存在する場合 :
SetFieldMapFromDDF DdfPath$, "Customers", "", "", _
    CustFldMap, False
DDF ファイルが存在しない場合 :
SetField CustFldMap(1), FLD_INTEGER, 4 'CustID
SetField CustFldMap(2), FLD_STRING, 24 'LastName
SetField CustFldMap(3), FLD_STRING, 24 'FirstName
SetField CustFldMap(4), FLD_STRING, 36 'Address1
SetField CustFldMap(5), FLD_STRING, 36 'Address2
SetField CustFldMap(6), FLD_STRING, 24 'City
SetField CustFldMap(7), FLD_STRING, 2 'State
SetField CustFldMap(8), FLD_STRING, 10 'Zip
SetField CustFldMap(9), FLD_STRING, 12 'HomePhone
SetField CustFldMap(10), FLD_STRING, 12 'WorkPhone
SetField CustFldMap(11), FLD_INTEGER, 4 'Status
SetField CustFldMap(12), FLD_BYTE, 1 'Member
SetField CustFldMap(13), FLD_INTEGER, 4 'Expiration

```

基本的なファイル操作を実行する

ファイルを開く

この操作を実行するには、ポジションブロック、ファイル名、オーナーの3つの情報が必要です。ポジションブロックとして、長さ 128 バイトのバイト配列変数を作成することにより、メモリを割り当てる必要があります。ファイル名は後ろにヌル文字を付けてキーバッファに格納し、オーナーも同様に後ろにヌル文字を付けてデータバッファに格納します。BTRCALL の `DataSize` パラメーターと `KeySize` パラメーターは、それぞれ最後に付けられたヌル文字も含む名前の長さに設定します。

- 1 ポジションブロックの変数を作成します。

```
' Customer テーブルを開く準備をする
Type posblk
    pbelements(0 To 127) As Byte
End Type
```

```
' ポジション ブロックとして 128 バイトの配列を定義する
Global CustPosBlk As posblk
```

- 2 次に、キーバッファ用のバイト配列を定義し、割り当てる必要があります。この配列は、キーバッファパラメーターの最大長または 255 バイトに設定することが推奨されます。

```
' キー バッファ長をバイト単位で定義 - 最大の 255 を定義
Global Const KEY_BUF_LEN = 255
```

```
' キー バッファのバイト配列を定義
Type kb_tmpb
```

```
    keybuff(1 To KEY_BUF_LEN) As Byte
End Type
Global KeyBufferByteArray As kb_tmpb
```

- 3 さらに、データバッファのバイト配列を定義して割り当てる必要があります。

```
' オーナー ネーム用のデータ バッファ長をバイト単位で定義
Global Const OWNER_MAX_LEN = 9
```

```
' データ バッファのバイト配列を定義
Type db_tmpb
```

```
    databuff(1 To OWNER_MAX_LEN) As Byte
End Type
Global db_datb As db_tmpb
```

- 4 次に、オーナーネームとパス文字列を保持する構造体を定義および設定する必要があります。

```

Public Type KeyBufferString
    FilePath As String * KEY_BUF_LEN
End Type
Global KeyBufferWorkingArea As KeyBufferString

Public Type OwnerDataBufferString
    OwnerName As String * OWNER_MAX_LEN
End Type
Global OpenFileOwnerDataBuffer As OwnerDataBufferString

```

5 パックおよびアンパックに必要なフィールド情報を保持する領域も定義する必要があります。

```

Global KeyBufferFldMap(0 To 0) As FieldMap
Global OpenFileDataBufferFldMap(0 To 0) As FieldMap

```

' これらのフィールドは次のように設定することができます。

```

SetField KeyBufferFldMap(0), FLD_STRING, KEY_BUF_LEN
SetField OpenFileDataBufferFldMap(0), FLD_STRING,
OWNER_MAX_LEN

```

6 キー バッファにテーブル名、OWNER にオーナー ネームを設定します。

```

KeyBufferWorkingArea.FilePath = ddfpath & "¥" &
"customer.mkd" & Chr$(0)

```

' オーナー ネームに空文字列を設定

```

OpenFileOwnerDataBuffer.OwnerName = "" & Chr$(0)

```

7 次に、これらの文字列を保持している構造体を、データ バッファおよびキー バッファバイト配列にコピーします。

```

StructToRow KeyBufferByteArray.keybuff,
KeyBufferFldMap, KeyBufferWorkingArea,
LenB(KeyBufferWorkingArea)
StructToRow db_datb.databuff,
OpenFileDataBufferFldMap, OpenFileOwnerDataBuffer,
LenB(OpenFileOwnerDataBuffer)

```

8 Open オペレーションを使用して BTRCALL オペレーションを呼び出します。

' Customer テーブルを開く

```

status = BTRCALL(BOPEN, CustPosBlk, db_datb, _
OWNER_MAX_LEN, KeyBufferByteArray, KEY_BUF_LEN, 0)

```

9 返されたステータスが正しいことを確認します。

' 返されたステータスを確認する

```
If status% <> 0 Then
    MsgBox "Customer テーブルのオープン エラー。Btrieve " & _
        " は次のステータスを返しました：" & status%
End If
```

ファイルを閉じる

ファイルを閉じる操作は簡単です。BCLOSE オペレーションを呼び出してポジションブロックを渡すのみです。ステータスが返され、この操作が正しく完了したかどうかを確認できます。

- 1 ポジションブロックを引数として Close オペレーションを呼び出します。

```
status% = BTRCALL(BCLOSE, CustPosBlk, 0, 0, 0, 0, 0)
```

- 2 返されたステータスを確認します。

```
If status% Then
    MsgBox "Customer テーブルのクローズ エラー。 " & _
        "Btrieve は次のステータスを返しました：" & status%
End If
```

ファイルを作成する

ファイルを作成するには、新しい Btrieve ファイルの作成に必要な情報を含む構造体を作成する必要があります。

- 1 必要な構造体および定数を含めます。

```
Type BtrFileSpec
    Length As Integer
    PageSize As Integer
    NumIndexes As Integer
    Reserved As Long
    FileFlags As Integer
    NumDupPtr As Byte
    NotUsed As Byte
    Allocation As Integer
End Type
```

’ ファイルのフラグを指定するために使用される定数 :

```
Global Const VAR_RECS = &h1
Global Const BLANK_TRUNC = &h2
Global Const PRE_ALLOC = &h4
Global Const DATA_COMP = &h8
Global Const KEY_ONLY = &h10
Global Const BALANCED_KEYS = &h20
Global Const FREE_10 = &h40
```

```

Global Const FREE_20 = &h80
Global Const FREE_30 = &hC0
Global Const DUP_PTRS = &h100
Global Const INCLUDE_SYSTEM_DATA = &h200
Global Const NO_INCLUDE_SYSTEM_DATA = &h1200
Global Const SPECIFY_KEY_NUMS = &h400
Global Const VATS_SUPPORT = &h800

```

- 2 ファイル仕様の構造体を作成して値を入力します。

```

Dim NewFileSpec As BtrFileSpec
' 長さ 100 のレコードを含むファイルを作成する
With NewFileSpec
    .Length = 100
    .PageSize = 4096
    .NumIndexes = 0
    .Reserved = 0
    .FileFlags = 0
    .NumDupPtr = 0
    .NotUsed = 0
    .Allocation = 0
End With

```

- 3 ファイル名を指定します。

```
KeyBufferWorkingArea.FilePath = "Example.btr" + Chr$(0)
```

- 4 構造体からパックされたキー バッファ バイト配列にコピーします。

```

StructToRow KeyBufferByteArray.keybuff,
    KeyBufferFldMap, KeyBufferWorkingArea,
    LenB(KeyBufferWorkingArea)

```

- 5 Btrieve Create 関数を呼び出します。

```

' Btrieve コマンドを呼び出してファイルを作成する
status% = BTRCALL(BCREATE, 0, NewFileSpec, 16, _
    KeyBufferByteArray, KEY_BUF_LEN, -1)

```

- 6 ステータスを確認します。

```

' 返されたステータスを確認する
If status% <> 0 Then
    MsgBox "ファイルの作成エラー。Btrieve " & _
        " は次のステータスを返しました：" & status%
End If

```

データの変更

レコードの挿入

挿入する行をデータ バッファに設定して BINSERT を呼び出すことにより、行の挿入が行えます。

- 1 構造体をパックされた行に変換します。

```
StructToRow custrow.buf, CustFldMap, custrec,  
LenB(keyrec)
```

- 2 Btrieve Insert 関数を呼び出します。

```
status% = BTRCALL(BINSERT, CustPosBlk$, custrow, _  
LenB(custrow), 0, 0, -1)
```

- 3 ステータスを確認します。

```
If status% <> 0 Then  
MsgBox "Insert エラー：" & status%  
End If
```

レコードの更新

行を更新するには、Get オペレーションまたは Step オペレーションを実行してカレンシーを確立する必要があります。完了後、その行は変更可能になります。

- 1 構造体をパックされた行に変換します。

```
StructToRow custrow.buf, CustFldMap, custrec,  
LenB(keyrec)
```

- 2 Btrieve Update 関数を呼び出します。

```
status% = BTRCALL(BUPDATE, CustPosBlk, custrow, _  
LenB(custrow), KeyBufferByteArray, KEY_BUF_LEN, 0)
```

- 3 ステータスを確認します。

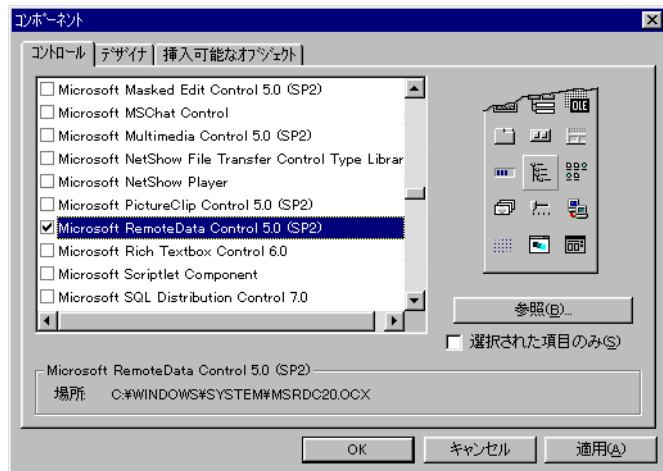
```
If status% <> 0 Then  
MsgBox "Update エラー：" & status%  
End If
```

レッスン 3 : Visual Basic で ODBC を使用する

SQL を使用して選択する

1 この例を実行するために必要なツールを選択します。

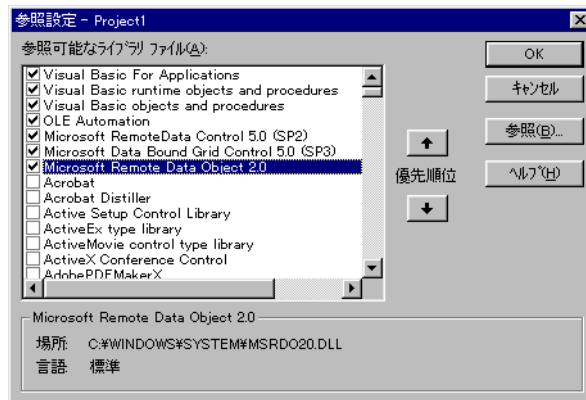
(このプログラムを実行するには、[プロジェクト | コンポーネント] で "Microsoft Data Bound Grid Control 5.0" および "Microsoft RemoteData Control 5.0" が選択されている必要があります。また、[プロジェクト | 参照設定] では、"Microsoft Remote Data Object 2.0"、"Remote Data Bound Grid Control 5.0"、"Microsoft RemoteData Control 5.0" がすべて選択されている必要があります。)



メモ Remote Data Control を使用するには、Microsoft Visual Basic Enterprise Edition 5.0 を使用する必要があります。

- 2 MSRDC コントロール、テキスト ボックス、DBGrid のいずれかをバインドします。MSRDC コントロールにバインドする場合は、RDO コントロールをツール ボックスに追加します。
 - a. [ツールボックス] ウィンドウを右クリックして [コンポーネント] をクリックします。
 - b. "Microsoft RemoteData Control 2.0" を選択して [OK] をクリックします。
 - c. [プロジェクト] をクリックし、次に [参照設定] をクリックします。

- d. "Microsoft Remote Data Object 2.0" を選択して [OK] をクリックします。



- 3 ツール ボックスの MSRDC コントロールをクリックします。次に、Form1 ウィンドウでカーソルをドラッグして MSRDC コントロールを作成します。
- 4 プロパティ ウィンドウの [Connect] フィールドに「database=Employees」と入力します。このフィールドでは、ODBC によるデータ アクセスのための接続文字列を設定し、接続先のテーブルを指定します。一覧からデータベースの種類を選択しなかった場合は、必然的に ODBC が使用されることになるため、このフィールドに、「ODBC」と入力する必要はありません。次に、[DataSourceName] フィールドに [Pvideo] と入力します。このフィールドでは、作成したデータソースの名前を指定します。[SQL] フィールドに次のクエリを入力します。

```
Select * from Employees
```

このクエリにより、Employee データベースからすべてのデータが抽出されます。

または、DB グリッドをバインドすることもできます。

- 1 [プロジェクト] の [コンポーネント] をクリックし、DBGrid をフォームに追加します。
- 2 プロパティ ウィンドウの [DataSource] フィールドを MSRDC コントロールの名前に設定します。
- 3 DBGrid を右クリックして [フィールドの取得] を選択します。このコマンドにより、MSRDC テーブル用に既に定義されている列名、長さ、データ型を使用して DBGrid が再定義されます。

MSRDC コントロール、テキスト ボックス、DBGrid のいずれかがバインドされた後、選択準備完了です。

例

```
Dim Qry As String
' クエリを作成する
Qry = "SELECT * FROM Employees"
Qry = Qry + " WHERE (LastName = '" + txtFind.Text + "')"
' データ コントロールをリフレッシュする
MSRDC1.SQL = Qry
MSRDC1.Refresh
DBGrid1.Refresh
```

SQL を使用して挿入する

BINSERT を呼び出すことにより、データ バッファにある行を挿入できます。

- 1 構造体をパックされた行に変換します。

```
StructToRow custrow.buf, CustFldMap, custrec,
LenB(keyrec)
```

- 2 Btrieve Insert 関数を呼び出します。

```
status% = BTRCALL(BINSERT, CustPosBlk$, custrow, _
LenB(custrow), 0, 0, -1)
```

- 3 ステータスを確認します。

```
If status% <> 0 Then
MsgBox "Insert エラー:" & status%
End If
```

MSRDC を使用して挿入する

テーブルに 1 つまたは複数のレコードを追加します。

- 1 テキスト ボックスを MSRDC コントロールにバインドします。

- a. テキスト ボックスをフォームに追加します。
- b. プロパティ ウィンドウの [DataSource] フィールドを MSRDC コントロールの名前に設定します。
- c. プロパティ ウィンドウの [DataField] フィールドをデータベースの任意のフィールド名に設定します。

例

```
' 一時的にこのテキストを保持する
text1temp$ = Text1.Text
text2temp$ = Text2.Text
text3temp$ = Text3.Text
```

- ・ テキスト ボックスがデータ ソースに
- ・ バインドされている場合、
- ・ バウンド コントロールに
- ・ データの変更を認識させるためには、
- ・ 次のコード行が必要です。

```
Text1.DataChanged = False
```

```
Text2.DataChanged = False
```

```
Text3.DataChanged = False
```

- ・ これらの初期値が既に存在しているか
- ・ チェックする必要があります。

SQL を使用して更新する

UPDATE ステートメントを使用することにより、データベース内の列の値を変更することができます。特定の列の値をすべて変更しない場合は、WHERE 句を使用してテーブル内で変更する行を定義することができます。

この例では、グリッド内の現在の Last Name を Text3 テキストボックス内のテキストに置き換えます。

- 1 この例を実行するために必要なツールを選択します。「この例を実行するために必要なツールを選択します。」を参照してください。
- 2 MSRD C コントロール、テキスト ボックス、DBGrid のいずれか、またはすべてをバインドします。

```
Dim strSQLChange As String
```

```
Dim qdfChange As rdoQuery
```

```
Dim rstEmployees As rdoResultset
```

- ・ テキスト ボックスがデータ ソースに

- ・ バインドされている場合、

- ・ バウンド コントロールに

- ・ コードの次の行が必要です。

```
Text3.DataChanged = False
```

- ・ グリッド内の Last Name を置き換える

```
Last$ = DBGrid1.Columns(2).Text
```

- ・ アクション クエリ用の SQL ステートメントを定義する

```
strSQLChange = "UPDATE Employees SET LastName = " & _  
"'" + Text3.Text + "' WHERE LastName = '" + Last$ + "'"
```

```
Set qdfChange = MSRD C1.Connection.CreateQuery("",  
strSQLChange)
```

```
Set rstEmployees = MSRD C1.Connection.OpenResultset( _  
"SELECT * FROM Employees", _  
dbOpenForwardOnly)
```

```
qdfChange.Execute
```

```

rstEmployees.Requery
MSRDC1.Refresh
rstEmployees.Close
DBGrid1.Refresh
' 新規レコードを追加する
MSRDC1.Resultset.AddNew
Text1.Text = text1temp$
Text2.Text = text2temp$
Text3.Text = text3temp$
'Resultset を更新する
MSRDC1.Resultset.Update

```

MSRDC を使用して更新する

データの更新後、次のコードを実行して変更を適用します。

例

```
MSRDC1.UpdateRow
```

パラメーター クエリを使用する

rdoQuery オブジェクトは、WHERE 句に 1 つまたは複数のパラメーターを持つ SQL クエリを実行するために使用されます。パラメーターは、実行されるたびに rdoQuery オブジェクトによって処理されます。これは、頻繁に使用するクエリを実行する際に役立ちます。パラメーター クエリは、ユーザーまたはアプリケーションによって指定されたパラメーターを通常のクエリに置き換えます。

パラメーター Insert ステートメントを使用する

- 1 この例を実行するために必要なツールを選択します。「この例を実行するために必要なツールを選択します。」を参照してください。
- 2 MSRDC コントロール、テキスト ボックス、DBGrid のいずれか、またはすべてをバインドします。「MSRDC コントロール、テキスト ボックス、DBGrid のいずれかをバインドします。MSRDC コントロールにバインドする場合は、RDO コントロールをツール ボックスに追加します。」を参照してください。
- 3 以下のコードを入力します。

```

Dim Qry As String
Dim qdfChange As rdoQuery
Dim rstEmployees As rdoResultset

```

```
' テキスト ボックスがデータ ソースに
' バインドされている場合、
' バウンド コントロールに
' コードの次の行が必要です。
Text1.DataChanged = False
Text2.DataChanged = False
Text3.DataChanged = False
' クエリを作成する
' パラメーターのプレースホルダーとして
' 疑問符を使用します。
Qry = "INSERT INTO Employees(Initials, LastName,
      FirstName) "
Qry = Qry + " VALUES (?, ?, ?)"
' クエリに名前を付けることにより、
' そのクエリに再度アクセスすることができますが、
' "" を使用した場合、このクエリは Query
' コレクションに追加されません。名前を付ければ、
' Query コレクションに追加されます。
Set qdfChange = MSRDC1.Connection.CreateQuery("", Qry)
' パラメーターを設定する
qdfChange.rdoParameters(0) = Text1.Text
qdfChange.rdoParameters(1) = Text2.Text
qdfChange.rdoParameters(2) = Text3.Text
Set rstEmployees = MSRDC1.Connection.OpenResultset( _
      "SELECT * FROM Employees", _
      dbOpenForwardOnly)
' SQL ステートメントを実行する
qdfChange.Execute
' データ コントロールをリフレッシュする
MSRDC1.Refresh
rstEmployees.Close
DBGrid1.Refresh
```

パラメーター Update ステートメントを使用する

- 1 この例を実行するために必要なツールを選択します。「この例を実行するために必要なツールを選択します。」を参照してください。
- 2 MSRDC コントロール、テキスト ボックス、DBGrid のいずれか、またはすべてをバインドします。「MSRDC コントロール、テキスト ボックス、DBGrid のいずれかをバインドします。MSRDC コントロールにバインドする場合は、RDO コントロールをツール ボックスに追加します。」を参照してください。

3 以下のコードを入力します。

```

Dim strSQLChange As String
Dim qdfChange As rdoQuery
Dim rstEmployees As rdoResultset
' テキスト ボックスがデータ ソースに
' バインドされている場合、
' バウンド コントロールに
' コードの次の行が必要です。
Text3.DataChanged = False
'DBGrid の 3 番目の列は LastName Column で、
' これは Update ステートメントの検索条件です。
Last$ = DBGrid1.Columns(2).Text
' アクション クエリ用の SQL ステートメントを定義する
' このステートメントは、グリッド内の LastName を
' Text3 内のテキストに置き換えます。
' パラメーターの場所に疑問符を使用します。
strSQLChange = "UPDATE Employees SET LastName = " & _
    "? WHERE LastName = '" + Last$ + "'"
' クエリに名前を付けることにより、
' そのクエリに再度アクセスすることができますが、
' "" を使用した場合、このクエリはコレクションに追加されません。
Set qdfChange = MSRDC1.Connection.CreateQuery("",
    strSQLChange)
' パラメーターを設定する
' これにより、Update ステートメント内の ? が置き換えられます。
qdfChange.rdoParameters(0) = Text3.Text
Set rstEmployees = MSRDC1.Connection.OpenResultset( _
    "SELECT * FROM Employees", _
    dbOpenForwardOnly)
' パラメーターを使用した SQL ステートメントを実行する
qdfChange.Execute
MSRDC1.Refresh
rstEmployees.Close

```


ActiveX を使用する Pervasive PSQL アプリケーションの作成

6

Visual Basic 5.0 と Pervasive PSQL ActiveX を使用してアプリケーションを作成する方法

このチュートリアルは、以下のセクションで構成されています。

- 「[ActiveX を使用した Visual Basic の例](#)」
- 「[レッスン 1 : データ ブラウザーを作成する](#)」
- 「[レッスン 2 : 更新用フォームを作成する](#)」
- 「[レッスン 3 : 2 つのテーブルからのレコードを結合する](#)」

ActiveX を使用した Visual Basic の例

このチュートリアルでは、Visual Basic および Pervasive による ActiveX インターフェイスを使用した Pervasive PSQL アプリケーションの作成方法を、順を追って説明します。ここでは、*file_path*¥PSQL¥Demodata¥ に収録されているデータ辞書ファイル (DDF) とデータファイルのサンプルを使用し、生徒情報を閲覧するためのアプリケーションを作成します。*file_path* はデフォルトで ¥Application Data¥Pervasive Software です (もしくは、¥ProgramData¥Pervasive Software)。

Pervasive PSQL ファイルのデフォルトの保存場所については、『Getting Started with Pervasive PSQL』の「[Pervasive PSQL ファイルがインストールされる場所](#)」を参照してください。

レッスン 1 : データ ブラウザーを作成する

このセクションでは、生徒情報を閲覧するためのフォームを作成し、Pervasive の ActiveX データ ソースとバウンド コントロールを、新しい Visual Basic プロジェクトに統合する方法を説明します。このレッスン終了後には、以下の操作が可能になります。

- 「VB プロジェクトへ Pervasive ActiveX インターフェイスへ追加する」
- 「VAccess コントロール フォームを作成する」
- 「Pervasive ActiveX インターフェイスのプロパティを設定する」
- 「バウンド コントロールのあるフォームを作成する」
- 「アプリケーションのテスト」

VB プロジェクトへ Pervasive ActiveX インターフェイスへ追加する

アプリケーションを作成するには、Visual Basic を起動し、新規に STANDARD.EXE プロジェクトを開始します。

Pervasive ActiveX インターフェイス使用の第一歩は、ActiveX コントロール コンポーネントを Visual Basic プロジェクトに追加することです。コンポーネントは、ACBTR732.OCX (Pervasive PSQL ActiveX データ ソース) および ACCTR732.OCX (Pervasive PSQL バウンド コントロール) の 2 つのファイルから構成されます。

➤Pervasive ActiveX インターフェイスを Visual Basic プロジェクトに追加するには

- 1 [プロジェクト] メニューから [コンポーネント] を選択します (またはツール ボックスを右クリックして [コンポーネント] を選択)。
- 2 [コントロール] リストをスクロールさせて "Pervasive ActiveX Data Source" および "Pervasive Bound Controls" チェック ボックスをオンにします。
- 3 [OK] をクリックします。

ツールボックスに Pervasive ActiveX アイコンが表示されます。



VAccess コントロール フォームを作成する

新規プロジェクトへの Pervasive ActiveX インターフェイスの追加後、プロパティ ウィンドウを使用して最初のフォームに名前を付けます。

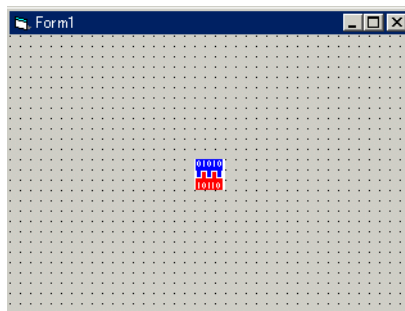
➤ フォームに名前を付けるには

- 1 Name プロパティに、"*frmVAccessForm*" を使用します。
- 2 Caption プロパティに、"*VAccess Form*" を使用します。

次は、フォームに VAccess コントロールを配置します。VAccess コントロールはデータをポイントするもので、ほかすべての Pervasive PSQL コントロールがバインドするマスター コントロールです。このコントロールは、Pervasive PSQL とのすべての通信を処理します。

➤ VAccess コントロールを Visual Basic フォームに追加するには

- 1 Pervasive PSQL ActiveX インターフェイスがある Visual Basic ツールボックスをクリックします。
- 2 VAccess コントロールのアイコンにマウスを置いて左クリックします。



- 3 フォームにカーソルを置いてドラッグし、コントロールを保持するための四角形を作成します。マウスを離すと、四角形の中に ActiveX コントロールが表示されます。

また、Pervasive PSQL の起動画面には、Pervasive PSQL データベース エンジンが起動されたことが表示されます。このコントロールは、設計時にはアイコンとして表示されますが、実行時には表示されません。

- 4 この VAccess コントロールの名前を、デフォルトのまま「*VAccess1*」にします。

プロジェクトのすべての VAccess コントロールは、バウンド コントロールとは別に、このフォームに配置します。VAccess コントロールを個々のフォームに配置する利点には、以下の 2 点が挙げられます。

- 作成時、データ コントロールが配置されるフォームが表示されている場合のみ、データ コントロールをほかのコントロールとバインドさせることができる（実行時は不可）。

ActiveX を使用する Pervasive PSQL アプリケーションの作成

- すべてのデータ コントロールには、配置されるフォームにかかわらず、独自の名前を付ける必要があるが、1つのフォームにまとめることにより、名前が重複する場合に Visual Basic の警告が表示される。

プロジェクトが拡大するにつれ、データ コントロールを1つのフォームにまとめる必要性が高まります。

Pervasive ActiveX インターフェイスのプロパティを設定する

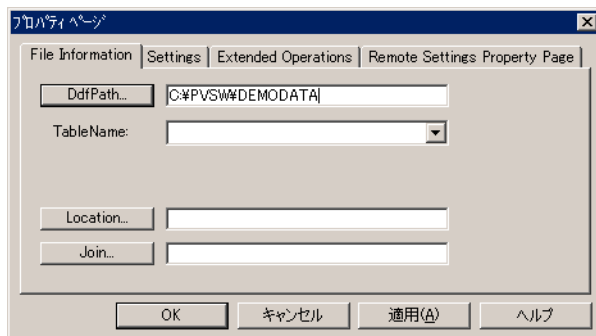
チュートリアルを続ける前に、このセクションでは、Pervasive PSQL ActiveX プロパティの設定方法を説明します。プロパティは、以下の3個所で設定できます。

- プロパティ ページ ダイアログ ボックス
- Visual Basic のプロパティ ウィンドウ
- Visual Basic コード エディター

プロパティ ページ

タブを含むこのダイアログ ボックスへは、コントロールを右クリックし、ショートカット メニューから [プロパティ] を選択してアクセスします。ここには、コントロールとデータへの接続に関するカスタム プロパティが表示されます。このチュートリアルでは、全体を通してこの [プロパティ ページ] ダイアログ ボックスを使用します。

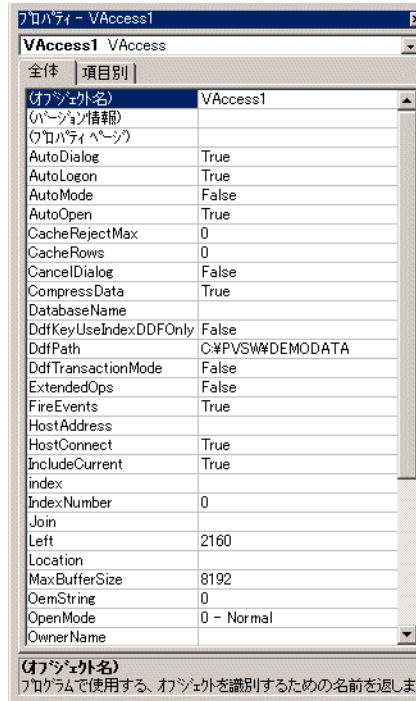
図 1 [プロパティ ページ] ダイアログ ボックス



Visual Basic のプロパティ ウィンドウ

ウィンドウには、選択したオブジェクトの全プロパティが表示されます。

図 2 Visual Basic のプロパティ ウィンドウ



Visual Basic コード エディター

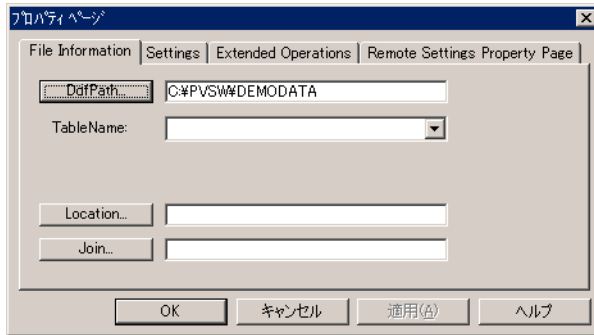
各プロパティの詳細については、『ActiveX Controls Guide』を参照してください。

これらのプロパティは、チュートリアル以下のセクションで設定します。

Pervasive PSQL ファイルへの接続

VAccess コントロールには、このコントロールを Pervasive PSQL データファイルへ接続するための重要なプロパティとして、DdfPath、TableName および Location という 3 つのプロパティがあります。これらのプロパティへは、コントロールを右クリックし、ショートカットメニューから [プロパティ] を選択して表示される [プロパティ ページ] から簡単にアクセスできます。また、プロパティ ウィンドウからもアクセス可能です。

図 3 データに接続するためのプロパティの設定



以下は、データへの接続に重要なプロパティです。

- **DdfPath** - このプロパティは、VAccess コントロールで使用する DDF (データ辞書ファイル) の場所を指定します。このプロパティでは、DDF が存在する有効なドライブおよびパスを指定します。
- **TableName** - このプロパティは、VAccess コントロールで使用する DDF のテーブル情報を指定します。このプロパティには、DdfPath で指定した DDF に含まれる有効なテーブル名を指定します。ドロップダウンメニューからテーブル名を選択できます。
- **Location** - このプロパティは、VAccess コントロールで使用する Pervasive PSQL データ ファイルを指定します。このプロパティは、FILE.DDF に含まれる Table Location に対応します。デフォルトでは、TableName プロパティで指定されたテーブルの Table Location になっていますが、データ ファイルへのパスを任意に指定することも可能です。

➤ プロパティ設定でデータを指定するには

- 1 デフォルトで、VAccess コントロールの DdfPath は `file_path\PSQL\DEMODATA` に設定されています (この設定はレジストリから読み取られます)。

Pervasive PSQL ファイルのデフォルトの保存場所については、『Getting Started with Pervasive PSQL』の「[Pervasive PSQL ファイルがインストールされる場所](#)」を参照してください。

- 2 **TableName** プロパティの使用可能な DDF が含まれるドロップダウンメニューから、**TableName** として "Student" を選択します。

すべての Pervasive PSQL バウンド コントロールは、この VAccess コントロールへのバインド時に、このテーブルを参照します。"Student" を選択すると、Location プロパティが STUDENT.MKD ファイルに変更されます。

ActiveX を使用する Pervasive PSQL アプリケーションの作成

この3つのファイルの場所プロパティは、以下のように設定されます。

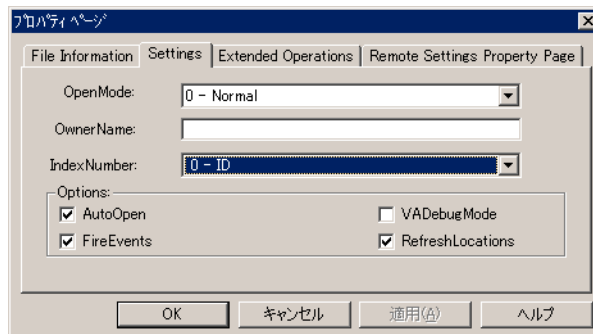
プロパティ	値
DdfPath	<i>file_path</i> ¥PSQL¥DEMOMDATA
TableName	Student
Location	<i>file_path</i> ¥PSQL¥DEMOMDATA¥STUDENT.MKD

ここで、インデックス番号を設定します。

➤ テーブルのインデックス番号を設定するには

- 1 *VAccess1* の [プロパティ ページ] ダイアログ ボックスの [Settings] タブを選択し、[IndexNumber] フィールドをクリックして Student テーブルに定義されているインデックスを表示します。
- 2 インデックス "0 - ID" を選択して [OK] をクリックします。図 4 に設定を示します。

図 4 IndexNumber プロパティの設定



バウンド コントロールのあるフォームを作成する

このセクションでは、Pervasive PSQL バウンド コントロールのフォームへの配置方法、および VAccess コントロールへのバインド方法を説明し、データベースの参照を可能にします。

➤ バウンド コントロールのあるフォームをセットアップするには

- 1 [プロジェクト] メニューの [フォーム モジュールの追加] をクリックします。
- 2 [開く] をクリックします (デフォルトではフォーム モジュールが選択されています)。

ActiveX を使用する Pervasive PSQL アプリケーションの作成

前セクションでの説明のように、すべてのバウンド コントロールは、VAccess コントロールとは別の個々のフォームに配置します。

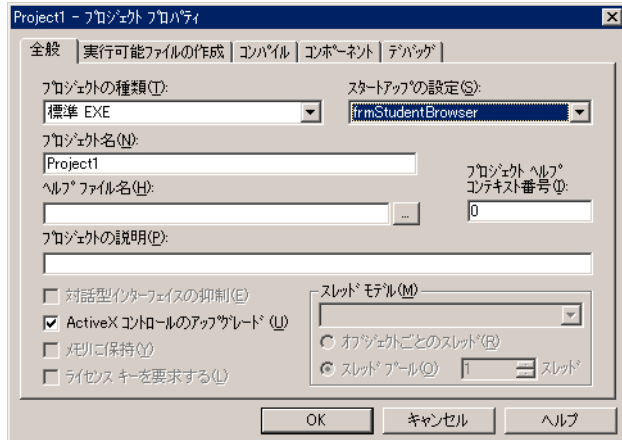
- 3 フォームのプロパティウィンドウで、Name プロパティを "*frmStudentBrowser*" に変更し、Caption プロパティを "*Pervasive PSQL Student Browser Sample Application*" に変更します。



メモ このチュートリアルでは、フォーム名の先頭に「*frm*」、コマンドコントロール名の先頭に「*cmd*」を使用します。これにより、プログラム作成に当たって、フォームとコマンドをほかのオブジェクトと区別します。その他のコントロールの名前には、デフォルト (VAText1 ボックスや VAText2 ボックスなど) を使用します。

- 4 [プロジェクト] メニューから [Project のプロパティ] を選択して [プロジェクト プロパティ] ウィンドウを開きます。[スタートアップの設定] を "*frmStudentBrowser*" に変更し、アプリケーション実行時にこのフォームが表示されるようにします。
- 5 [OK] をクリックします。

図 5 スタートアップのフォームの設定



- 6 次に、*frmStudentBrowser* フォームをダブルクリックして `Form_Load` イベントを開きます。このイベントに、以下のコードを入力します。

```
frmVAccessForm.VAccess1.GetFirst
```

このコードでは、`GetFirst` メソッドを使用して *frmVAccessForm* の *VAccess1* (VAccess コントロール) に、最初のレコードを取得するように指示します。これにより、フォームに配置するバウンド コントロールが、最初のアプリケーション実行時にデータを表示します。

インデックス検索を追加する

次に、バウンド コントロールを *frmStudentBrowser* フォームに配置します。最初に、テーブル インデックスの検索に使用するテキスト ボックスを追加します。

▶ テキスト ボックスを追加するには

- 1 フォームが表示されていない場合は、*frmStudentBrowser* フォームに移動します。
- 2 フォームの左上の角に VAtext ボックスを配置します。

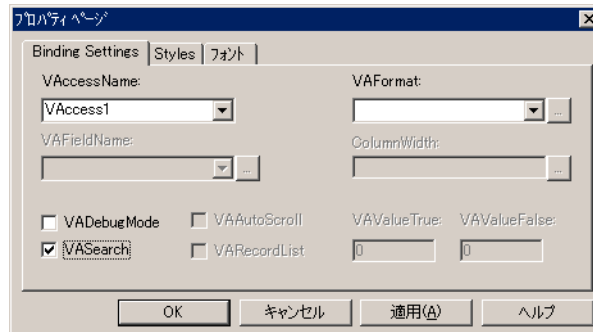


生徒 ID の 9 桁を入力できるようにボックスのサイズを変更します。

- 3 テキスト ボックスを右クリックして [プロパティ] を選択し、[プロパティ ページ] を表示します。
- 4 VAccessName のプロパティに、メニューから "VAccess1" を選択します。これにより、VAtext ボックスが *VAccess1* (VAccess コントロール) にバインドされます。
- 5 VASearch チェック ボックスをオンにします。

この設定により、特定のフィールドを対象とした現在のインデックスの検索が可能になります。*VAccess1* コントロールに選択されたインデックスが自動的に使用されるため、VAFieldName は必要ありません。

図 6 テキスト ボックスの検索インデックスの設定



- 6 [OK] をクリックします。
- 7 必要に応じて Visual Basic Label コントロールをテキスト ボックス左側に配置し、*Caption* プロパティを以下のように変更します。

「生徒レコードを指定する値を入力してください。」

リスト ボックスを追加する

VAList ボックスは、データベースからの情報のフィールドを、リスト形式で表示する場合に使用できます。

▶ リスト ボックスを追加するには



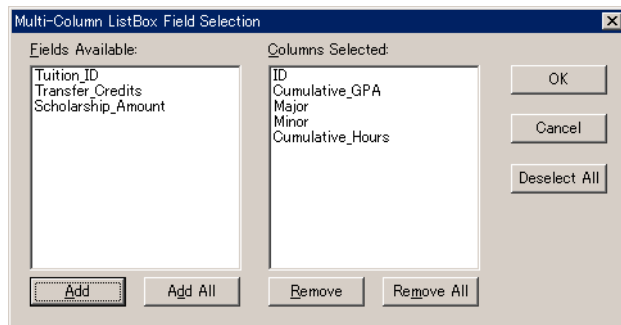
- 1 VAList ボックスをフォームに配置し、その VAccessName プロパティを "VAccess1" に変更することにより、VAccess コントロールにバインドします。



メモ 複数のデータ行が表示できるようリスト ボックスのサイズを変更します。リスト ボックス自体が小さすぎる場合、手順 5 で行のサイズを効率的に変更することができなくなります。

- 2 [プロパティ ページ] ダイアログ ボックスの [VARecordList] チェック ボックスをオンにします。これにより、このプロパティが True に設定され、異なる行へのフィールドの表示が可能になります。
- 3 [プロパティ ページ] ダイアログ ボックスの VAFieldName プロパティの隣の [...] ボタンをダブルクリックし、使用可能なフィールドが表示されたダイアログ ボックスを開きます。
- 4 複数の行を作成する場合は、使用可能なフィールドをダブルクリックして選択します (またはクリックして選択した後 [追加] をクリック)。図 7 に示されているように、フィールドを選択します。ID、Cumulative_GPA、Major、Minor、Cumulative_Hours の各タイトルを使用します。[OK] をクリックします。

図 7 リスト ボックス行の選択



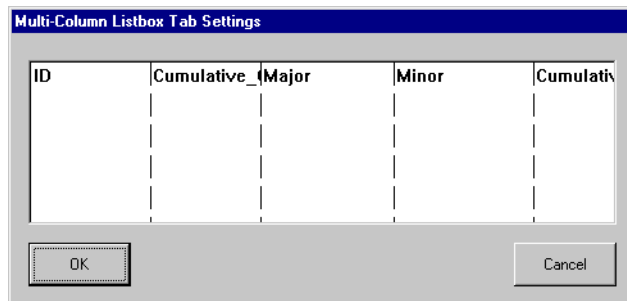
- 5 行にデータが順番に表示されるように `ColumnWidth` プロパティを調整します。調整は、`[ColumnWidth]` コンボ ボックスの隣の `[...]` ボタンをクリックし、`[Multi-Column Tab Settings]` ウィンドウを表示して行います。選択した最初のフィールド `[ID]` が表示されます。



メモ すべての行が表示されない場合は、このウィンドウと [プロパティ ページ] を閉じ、リスト ボックスのサイズを変更して再試行します。

- 6 `[ID]` の右側をダブルクリックして最初のタブ ストップを設定します。破線が現れます。この線をドラッグし、この列に表示されるデータ用のスペースを確保します。ダブルクリックとドラッグを繰り返し、4 つの行のタブを表示および調節します。
- 7 終了後、ウィンドウは、図 8 のようになります。このウィンドウの `[OK]` をクリックし、`[プロパティ ページ]` の `[OK]` をクリックして変更を適用します。

図 8 リスト ボックス行のサイズ変更



- 8 5 つの Visual Basic Label コントロールをフォームの `VAlist` ボックスの上に配置し、各行にラベルを付けます。`ID`、`Cumulative_GPA`、`Major`、`Minor`、`Cumulative_Hours` の各タイトルを使用します。

縦スクロール バーを追加する



`VAVScrollBar` により、データベースの閲覧が容易になります。

➤ 縦スクロール バーを追加するには

- 1 `VAVScrollBar` をフォームに配置し、縦スクロール バーとして `VAlist` ボックス右側に設置します。

- 2 スクロール バーの VAccessName プロパティを "VAccess1" に設定し、VAccess コントロールにバインドします。

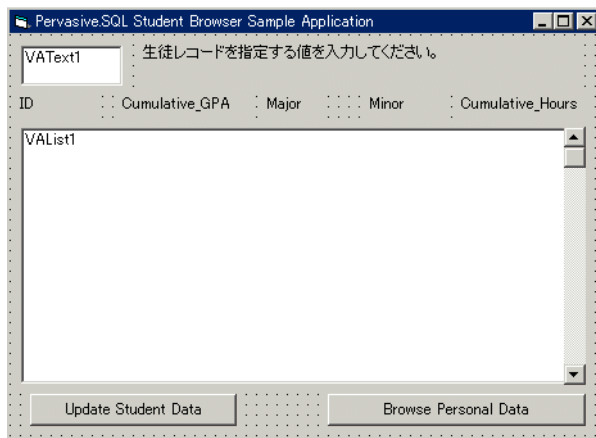
Visual Basic コマンド ボタンを追加する

Visual Basic のコマンド ボタンを使用し、ほかのフォーム（後述）を呼び出すことができます。

▶VB コマンド ボタンを追加するには

- 1 2つの Visual Basic コマンド ボタンをフォーム下部に配置します。
- 2 1つのボタンの Name プロパティを "cmdUpdate" に、もう 1つのボタンの Name プロパティを "cmdBrowse" に変更します。
- 3 1つのボタンの Caption プロパティを "Update Student Data" に、もう 1つのボタンの Caption プロパティを "Browse Personal Data" に変更します。これらのボタンを機能させるには、コードを追加（後述）します。フォームは、図 9 のようになります。

図 9 実行前の Student Browser



アプリケーションのテスト

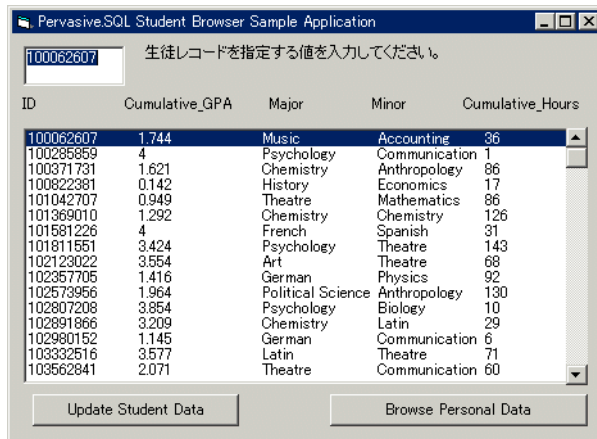
➤ この時点でアプリケーションをテストするには

- 1 まず、[ファイル] メニューの [名前をつけてプロジェクトの保存] から保存を行います。
- 2 [実行] メニューの [開始] をクリックします。

VAText ボックスと VAList ボックスが同期し、テキスト ボックスに生徒 ID を入力すると、リスト ボックス上部にそれが表示されます。また、VAVScrollBar を使用してデータベースをスクロールすることもできます。

必要に応じて ColumnWidth プロパティから、VAList ボックスのサイズ、行ラベル コントロール、タブの位置を変更してフォームのデザインを統一します。コンパイル後のサンプル ブラウザーは、図 10 のようになります。

図 10 実行後の Student Browser



レッスン 2 : 更新用フォームを作成する

このセクションでは、プロジェクトへのフォーム追加方法を説明します。このフォームは、**Student Browser** で情報の更新、生徒の補足情報の表示、生徒レコードの削除を行うために使用します。このフォームでは、複数の **Pervasive PSQL ActiveX** テキスト ボタン、オプション ボタン、コマンド ボタンを使用します。

このレッスン終了後には、以下の操作が可能になります。

- 「[フォームの追加と表示](#)」
- 「[複数のテキスト ボックスを配置する](#)」
- 「[コマンド ボタンを追加する](#)」

フォームの追加と表示

➤ プロジェクトにフォームを追加するには

- 1 [プロジェクト] メニューの [フォーム モジュールの追加] をクリックします。[開く] をクリックします

Visual Basic では、プロジェクトに 3 つ目のフォームが追加されます。

- 2 フォームの Name プロパティを "*frmStudentUpdate*" に、Caption プロパティを "*Update Student Information*" に変更します。

[Update Student Data] ボタンを押すと、*frmStudentBrowser* フォームに *frmStudentUpdate* フォームが表示されます。特定の生徒 ID を選択して [Update Student Data] ボタンをクリックした場合は、その生徒の詳細がフォームに表示されます。

- 3 *frmStudentBrowser* フォームに切り換え、[Update Student Data] ボタンをダブルクリックしてコード ウィンドウにそのコードを表示します。
- 4 [Update Student Data] ボタンの Click イベントに以下のコードを入力します。

```
frmStudentUpdate.Show  
frmVAccessForm.VAccess1.Refresh
```

最初の行ではフォームが表示され、次の行で *VAccess1* (*VAccess* コントロール) にバインドされたすべてのコントロールのデータを更新する **Refresh** メソッドが呼び出されます。ここで **Refresh** メソッドが使用されない場合、*frmStudentUpdate* フォームに配置したコントロールは、フォームが表示された際に空白になります。**Refresh** メソッドは、*VAccess* コントロールのデータ バッファーに応じてこれらのコントロールのデータを更新し、フォーム表示の際にデータが表示されます。

複数のテキスト ボックスを配置する

➤ VAMText ボックス を追加するには

- 1 8つの VAMText ボックスを *frmStudentUpdate* フォームに追加して縦の行に配置します。



- 2 プロパティ ウィンドウから、各テキスト ボックスの VAMAccessName プロパティを "*VAMAccess1*" に設定します。
- 3 各 VAMText ボックスの VAMFieldName プロパティを、以下の順番でいずれかの名前に設定します。
 - ID
 - Tuition_ID
 - Transfer_Credits
 - Scholarship_Amount
 - Cumulative_Hours
 - Cumulative_GPA
 - Major
 - Minor

コマンド ボタンを追加する

➤ コマンド ボタンを追加するには

- 1 3つの VAMCommandButtons ボタンを *frmStudentUpdate* フォームに配置して、各 Name プロパティには "Insert"、"Update"、"Delete" を、各 Caption プロパティには "*cmdInsert*"、"*cmdUpdate*"、"*cmdDelete*" を設定します。



- 2 すべてのボタンの VAMAccessName プロパティを "*VAMAccess1*" に設定します。ほかのコントロールと同様に、このプロパティを設定することにより、コントロールと VAMAccess コントロールがバインドされます。
- 3 [Insert] ボタンの VAMOperation プロパティを "*2-Insert*" に、[Update] ボタンの VAMOperation プロパティを "*3-Update*" に、[Delete] ボタンの VAMOperation プロパティを "*4-Delete*" に設定します。

VAMOperation プロパティは、ボタンのクリック時に実行される Pervasive PSQL の処理を VAMCommandButton に割り当てるためのものです。この処理は、Click イベントの任意コードの後に実行されます。

ActiveX を使用する Pervasive PSQL アプリケーションの作成

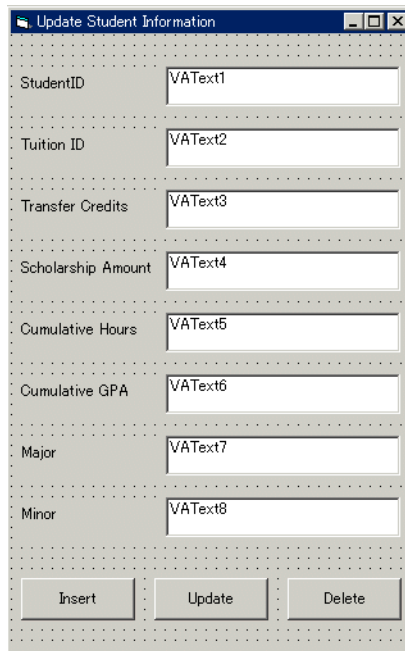
以上で、新規生徒レコードのデータベースへの追加、既存レコードの変更、レコードの削除を行う各ボタンが、フォームに追加されました。



メモ 多くのフィールドは、両方のフォームとも同様ですが、互いに独自のフィールドがあります。共通のフィールドは、ユーザーによる更新が可能なものです。

必要に応じて、Visual Basic Label コントロールを使用してフォームの VAMText ボックスを識別し、フォーム上のコントロールを整頓します。フォームは以下の図のようになります。

図 11 生徒情報更新フォームの完成版

A screenshot of a Windows-style application window titled "Update Student Information". The window contains a form with eight text input fields, each with a label to its left: "StudentID" (VA Text1), "Tuition ID" (VA Text2), "Transfer Credits" (VA Text3), "Scholarship Amount" (VA Text4), "Cumulative Hours" (VA Text5), "Cumulative GPA" (VA Text6), "Major" (VA Text7), and "Minor" (VA Text8). At the bottom of the form are three buttons: "Insert", "Update", and "Delete". The form is set against a dotted grid background.

アプリケーションのテスト

➤ このレッスンで追加した機能をテストするには

- 1 保存後、アプリケーションを実行します。
- 2 *frmStudentBrowser* フォームのレコードをスクロールし、[Update Student Data] ボタンをクリックします。

[*Update Student Information*] フォームが開き、リストで選択した生徒に関するフィールドが表示されます。[*Student Browser*] ウィンドウのリストをスクロールすると、両方のフォームに表示されているデータが同期します。

- 3 [*Update Student Information*] フォームの VAMText ボックスに新しいデータを入力し、[Insert]、[Update]、[Delete] をクリックしてそれぞれの操作をレコードに対して行います。

レッスン 3 : 2 つのテーブルからのレコードを結合する

このレッスンでは、関連する 2 つの異なるテーブルからのレコードを表示する方法を説明します。これは、データベースのプログラミングに便利なテクニックです。このレッスンでは、*Student* テーブルの生徒の成績情報に関連する *Person* テーブルからの同一生徒の個人情報を表示します。ここでは、以下の操作を行います。

- 「VAccess コントロールの追加」
- 「VAccess コントロールの結合」
- 「個人データ フォームの追加と表示」
- 「アプリケーションのテスト」

VAccess コントロールの追加

➤ VAccess コントロールを追加するには

- 1 2 つ目の VAccess コントロールを *frmVAccess* フォームに配置し、Name プロパティを "*VAccess2*" のままにします。
- 2 3 つのファイルの場所プロパティを以下のように設定します。

プロパティ	値
DdfPath	<i>file_path</i> ¥PSQL¥DEMOMDATA
TableName	Person
Location	<i>file_path</i> ¥PSQL¥DEMOMDATA¥Person.mkd

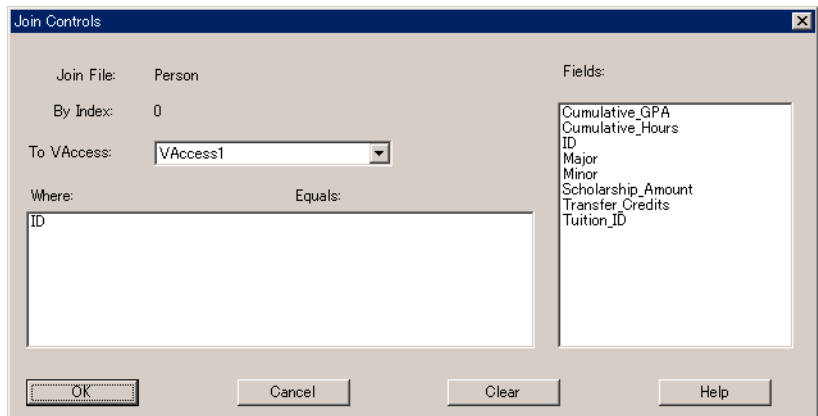
Pervasive PSQL ファイルのデフォルトの保存場所については、『Getting Started with Pervasive PSQL』の「[Pervasive PSQL ファイルがインストールされる場所](#)」を参照してください。

VAccess コントロールの結合

➤2 つの VAccess コントロールを結合するには

- 1 [プロパティ ページ] ダイアログ ボックスの [Join] をクリックし、次のダイアログ ボックスを開きます。

図 12 [Join Controls] ダイアログ ボックス



このダイアログ ボックスに表示される情報には、結合するテーブル (*Person*)、そのテーブルで選択されているインデックス (*0-ID*)、結合する VAccess コントロール (*VAccess1*)、コントロールで使用可能なフィールドが含まれます。

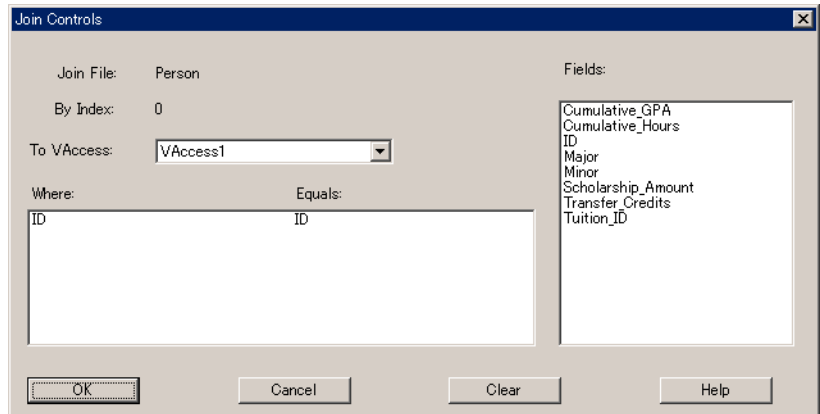
2 つのフィールドを結合するには、各 VAccess コントロールのデータ型が同じ (両方が文字列、両方が整数など) である必要があります。この場合、結合するフィールドの名前も同じ (*ID* と *ID*) ですが、これを同じにする必要はありません。

左側のリスト ボックスは、*Person* テーブルのインデックス (*ID*) です。このフィールドのデータは、*VAccess1* (VAccess コントロール) の *ID* フィールドと同等のもので、右側のリスト ボックスの 3 番目の項目です。

- 2 [*Fields*] リスト内の "*ID*" をダブルクリックし、*VAccess2* コントロールの *ID* フィールドを、*VAccess1* コントロールの *ID* フィールドに結合します。

"*ID*" フィールドが [*Equals*] 行に表示され、*VAccess2* コントロールの *ID* フィールドに結合されます。図 13 のようなダイアログ ボックスが表示されます。

図 13 2つの VAccess コントロールの結合フィールド



- 3 [OK] をクリックします。[プロパティ ページ] ダイアログ ボックスにある [Join] ボタンの隣のテキスト ボックスに、以下のテキストが表示されます。

VAccess1:ID

このテキストは、選択した VAccess2 コントロールが、ID フィールドを通じて VAccess1 コントロールに結合されることを表します。

- 4 [OK] をクリックして結合を実行します。

個人データ フォームの追加と表示

ここでは、別のフォームを追加します。このフォームには、生徒の個人データが読み取り専用で表示されます。

➤ 結合フィールドのある読み取り専用フォームを作成するには

- 1 プロジェクトにフォームを追加します。Name プロパティを "frmPersonalData" に、Caption プロパティを "Personal Data" に変更します。
- 2 frmStudentBrowser フォームの [Browse Personal Data] ボタンのクリック時に PersonalData フォームを表示させるには、フォームに移動してボタンをダブルクリックします。Click イベントに以下のコードを入力します。

```
frmPersonalData.Show
```

```
VAccessForm.VAccess2.Refresh
```

- 3 次に、下図を参考にテキスト ボックスとラベルを追加し、後に追加する [Male/Female] オプション ボタン用のスペースを確保しておきます。

図 14 読み取り専用個人データ フォーム

The screenshot shows a window titled "Personal Data" with a grid background. It contains several text input fields: "Last Name" (VA_Text1), "First Name" (VA_Text2), "Street" (VA_Text3), "City" (VA_Text4), "State" (VA_Text5), and "Zip" (VA_Text6). There is also a "Sex" section with two radio buttons labeled "Male" and "Female".

- 4 各コントロールの VAccessName プロパティを "VAccess2" に設定し、以下の VAFieldName をそれぞれ選択します。
 - Last_Name
 - First_Name
 - Perm_Street
 - Perm_City
 - Perm_State
 - Perm_Zip
- 5 各テキスト ボックスの [プロパティ ページ] で、[Styles] タブをクリックします。[ReadOnly] チェック ボックスをオンにして [OK] をクリックします。

図 15 テキスト ボックスの読み取り専用設定

The screenshot shows the "Properties" dialog box with the "Styles" tab selected. The "ReadOnly" checkbox is checked. Other options include "Password", "FocusSelText", "MultiLine", and "WantReturn", all of which are unchecked. The "ScrollBars" dropdown is set to "None". The "Max Length" field is set to "0".

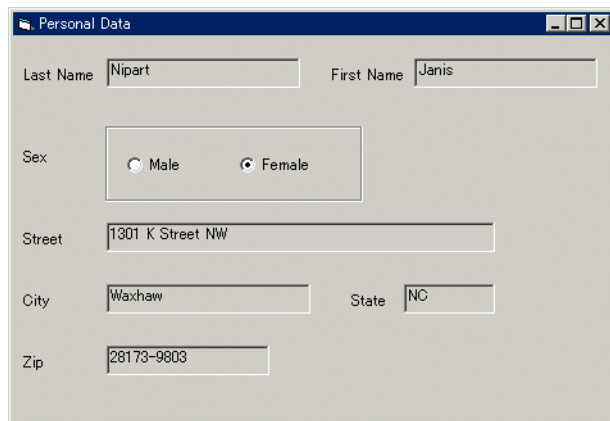
- 6 [Male/Female] オプション コントロールを配置する Visual Basic Frame コントロールを、フォームに作成します。
- 7 2つの VAOptionButton をフレーム内に追加し、[Male] および [Female] ラベルを追加します。

ActiveX を使用する Pervasive PSQL アプリケーションの作成

- 8 両方のオプション ボタンの [プロパティ ページ] で、VaccessName に "VAccess2" を、VAFieldName に "Sex" を選択します。
- 9 [Female] ボタンに [Male] ボタンと逆の情報を表示するには、VAValueTrue を 0 に、VAValueFalse を 1 に変更します。
- 10 [Male/Female] ボタンを読み取り専用にするには、Frame の Enabled プロパティを False に設定します。

frmStudentBrowser フォームではほかの生徒 ID が選択された場合、その生徒の情報は *Personal Data* フォームに表示されます。

図 16 コンパイル後の個人データ フォーム



The screenshot shows a window titled "Personal Data" with the following fields and values:

Field	Value
Last Name	Nipart
First Name	Janis
Sex	Female (selected)
Street	1301 K Street NW
City	Waxhaw
State	NC
Zip	28173-9803

アプリケーションのテスト

保存後、再びアプリケーションのテストを行います。プログラムを実行し、アプリケーションの全機能をテストします。

Btrieve API の言語インターフェイス

7

この章では、以下の言語インターフェイスについて説明します。

- 「C/C++」
- 「COBOL」
- 「Delphi」
- 「Pascal」
- 「Visual Basic」

C/C++

このセクションでは、以下について説明します。

- 「[サンプルプログラムのコンパイル、リンク、実行](#)」
- 「[C++ Builder アプリケーションのコンパイル](#)」

C++ プログラムの例

Web ダウンロードにより提供される以下のプログラム例では、Btrieve の一般的な操作方法を説明します。これらの操作は、MicroKernel との依存関係で要求される順番（ファイルを開いてから I/O を実行するなど）で行われます。

たとえば、Btrieve Login オペレーションの例については、「[Btrieve Login の C++ プログラムの例](#)」を参照してください。

Btrsamp.c

```
/*
*****
Copyright 2003 Pervasive Software Inc. All Rights Reserved
*****
*/
BTRSAMP.C
```

これは、目的の環境で、コンパイラ ツールを使用した Btrieve アプリケーションのコンパイル、リンク、実行が可能なことを確認するための簡単なサンプルです。

このプログラムでは、Btrieve の Windows 用 C/C++ インターフェイスを示します。

このプログラムでは、サンプル データベースで以下の操作を行います。

- Microkernel Database エンジンのバージョン情報を取得する
- sample.btr を開く
- Key 0 の既知の値を持つレコードを取得する
- 読み込んだレコードを表示する
- Stat オペレーションを実行する
- 空の sample.btr のクローンを作成して開く
- Get Next Extended を実行して sample.btr のレコードのサブセットを抽出する
- クローン ファイルにこれらのレコードを挿入する
- 両ファイルを閉じる

重要: Btrieve データ ファイルのサンプル sample.btr が保存されているディレクトリへのフルパスを指定してください。また、それぞれの「重要」項目も参考にしてください。

このプログラムは、インターフェイス モジュールが対応するプラットフォームで実行できます。プラットフォームは btrapi.h にリストされているプラットフォームスイッチで表されます。Windows では、コンソール アプリケーションで標準出力

printf() を使用できるようにしてください。多くの C/C++ コンパイラは、標準 I/O を使用した Windows アプリケーションに対応しています。

アプリケーションのターゲット プラットフォーム選択に関する情報は、btrapi.h の始まりを参照してください。ターゲット プラットフォームを指定してください。

```

*****/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <btrapi.h>
#include <btrconst.h>

/*****
  定数
*****/
/*****
  重要 : 以下を変更し、使用する sample.btr へのフルパスを指定してください。
*****/
#ifdef BTI_WIN_32
#define FILE1_NAME
#endif
"<path>¥¥samples¥¥sample.btr"

#ifdef BTI_WIN_32
#define FILE2_NAME "<path>¥¥samples¥¥sample2.btr"
#endif

#define EXIT_WITH_ERROR      1
#define TRUE                 1
#define FALSE                0
#define VERSION_OFFSET      0
#define REVISION_OFFSET     2
#define PLATFORM_ID_OFFSET  4
#define MY_THREAD_ID        50

/* 弊社の構造体に定義を加えないでください。*/
#if defined(__BORLANDC__)
  #pragma option -a-
#else
  #if defined(_MSC_VER) || defined(__WATCOMC__)
    #pragma pack(1)
  #endif
#endif
#endif

/*****
  クライアント ID とバージョンの構造体の型定義
*****/
typedef struct
{

```

Btrieve API の言語インターフェイス

```
BTI_CHAR  networkAndNode[12];
BTI_CHAR  applicationID[2];
BTI_WORD  threadID;
} CLIENT_ID;

typedef struct
{
    BTI_SINT  Version;
    BTI_SINT  Revision;
    BTI_CHAR  MKDEId;
} VERSION_STRUCT;

/*****
sample.btr からのレコードの定義
*****/
typedef struct
{
    BTI_LONG  ID;
    BTI_CHAR  FirstName[16];
    BTI_CHAR  LastName[26];
    BTI_CHAR  Street[31];
    BTI_CHAR  City[31];
    BTI_CHAR  State[3];
    BTI_CHAR  Zip[11];
    BTI_CHAR  Country[21];
    BTI_CHAR  Phone[14];
} PERSON_STRUCT;

/*****
Stat/Create 構造体の型定義
*****/
typedef struct
{
    BTI_SINT  recLength;
    BTI_SINT  pageSize;
    BTI_SINT  indexCount;
    BTI_CHAR  reserved[4];
    BTI_SINT  flags;
    BTI_BYTE  dupPointers;
    BTI_BYTE  notUsed;
    BTI_SINT  allocations;
} FILE_SPECS;

typedef struct
{
    BTI_SINT  position;
    BTI_SINT  length;
    BTI_SINT  flags;
    BTI_CHAR  reserved[4];
    BTI_CHAR  type;
}
```

```

    BTI_CHAR    null;
    BTI_CHAR    notUsed[2];
    BTI_BYTE    manualKeyNumber;
    BTI_BYTE    acsNumber;
} KEY_SPECS;

typedef struct
{
    FILE_SPECS  fileSpecs;
    KEY_SPECS   keySpecs[5];
} FILE_CREATE_BUF;

/*****
    Get Next Extended 処理用の構造体の型定義
*****/
typedef struct
{
    BTI_SINT    descriptionLen;
    BTI_CHAR    currencyConst[2];
    BTI_SINT    rejectCount;
    BTI_SINT    numberTerms;
} GNE_HEADER;

typedef struct
{
    BTI_CHAR    fieldType;
    BTI_SINT    fieldLen;
    BTI_SINT    fieldOffset;
    BTI_CHAR    comparisonCode;
    BTI_CHAR    connector;
    BTI_CHAR    value[3];
} TERM_HEADER;

typedef struct
{
    BTI_SINT    maxRecsToRetrieve;
    BTI_SINT    noFieldsToRetrieve;
} RETRIEVAL_HEADER;

typedef struct
{
    BTI_SINT    fieldLen;
    BTI_SINT    fieldOffset;
} FIELD_RETRIEVAL_HEADER;

typedef struct
{
    GNE_HEADER    gneHeader;
    TERM_HEADER   term1;
    TERM_HEADER   term2;
}

```

Btrieve API の言語インターフェイス

```
    RETRIEVAL_HEADER      retrieval;
    FIELD_RETRIEVAL_HEADER recordRet;
} PRE_GNE_BUFFER;

typedef struct
{
    BTI_SINT      recLen;
    BTI_LONG      recPos;
    PERSON_STRUCT personRecord;
} RETURNED_REC;

typedef struct
{
    BTI_SINT      numReturned;
    RETURNED_REC recs[20];
} POST_GNE_BUFFER;

typedef union
{
    PRE_GNE_BUFFER preBuf;
    POST_GNE_BUFFER postBuf;
} GNE_BUFFER, BTI_FAR* GNE_BUFFER_PTR;

/* 構造体パッキングの復元 */
#if defined(__BORLANDC__)
    #pragma option -a.
#else
    #if defined(_MSC_VER) || defined(__WATCOMC__)
        #pragma pack()
    #endif
#endif
#endif

/*****
    メイン
*****/
int main(void)
{
    /* Btrieve 関数パラメーター */
    BTI_BYTE posBlock1[128];
    BTI_BYTE posBlock2[128];
    BTI_BYTE dataBuf[255];
    BTI_WORD dataLen;
    BTI_BYTE keyBuf1[255];
    BTI_BYTE keyBuf2[255];
    BTI_WORD keyNum = 0;

    BTI_BYTE btrieveLoaded = FALSE;
    BTI_LONG personID;
    BTI_BYTE file1Open = FALSE;
    BTI_BYTE file2Open = FALSE;
}
```

```

BTI_SINT status;
BTI_SINT getStatus = -1;
BTI_SINT i;
BTI_SINT posCtr;

CLIENT_ID      clientID;
VERSION_STRUCT versionBuffer[3];
FILE_CREATE_BUF fileCreateBuf;
GNE_BUFFER_PTR gneBuffer;
PERSON_STRUCT  personRecord;

printf("***** Btrieve C/C++ Interface Demo
*****\n\n");

/* クライアント ID のセットアップ */
memset(clientID.networkAndNode, 0, sizeof(clientID.networkAndNode));
memcpy(clientID.applicationID, "MT", 2); /* "AA" 以上を設定 */
clientID.threadID = MY_THREAD_ID;

memset(versionBuffer, 0, sizeof(versionBuffer));
dataLen = sizeof(versionBuffer);

status = BTRVID(
    B_VERSION,
    posBlock1,
    &versionBuffer,
    &dataLen,
    keyBuf1,
    keyNum,
    (BTI_BUFFER_PTR)&clientID);
if (status == B_NO_ERROR)
{
    printf("Btrieve Versions returned are:\n");
    for (i = 0; i < 3; i++) {
        if (versionBuffer[i].Version != 0)
        {
            printf("    %d.%d %c\n", versionBuffer[i].Version,
                versionBuffer[i].Revision,
                versionBuffer[i].MKDEID );
        }
    }
    printf("\n");
    btrieveLoaded = TRUE;
}
else
{
    printf("Btrieve B_VERSION status = %d\n", status);
    if (status != B_RECORD_MANAGER_INACTIVE)
    {
        btrieveLoaded = TRUE;
    }
}

```

Btrieve API の言語インターフェイス

```
    }
}

/* バッファークリア */
if (status == B_NO_ERROR)
{
    memset(dataBuf, 0, sizeof(dataBuf));
    memset(keyBuf1, 0, sizeof(keyBuf1));
    memset(keyBuf2, 0, sizeof(keyBuf2));
}

/* sample.btr を開く */
if (status == B_NO_ERROR)
{
    strcpy((BTI_CHAR *)keyBuf1, FILE1_NAME);
    strcpy((BTI_CHAR *)keyBuf2, FILE2_NAME);

    keyNum = 0;
    dataLen = 0;

    status = BTRVID(
        B_OPEN,
        posBlock1,
        dataBuf,
        &dataLen,
        keyBuf1,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_OPEN status (sample.btr) = %d\n", status);
    if (status == B_NO_ERROR)
    {
        file1Open = TRUE;
    }
}

/* B_GET_EQUAL を使用して key 0 = 263512477 のレコードを取得 */
if (status == B_NO_ERROR)
{
    memset(&personRecord, 0, sizeof(personRecord));
    dataLen = sizeof(personRecord);
    personID = 263512477; /* 実際これは社会保障番号 */
    *(BTI_LONG BTI_FAR *)&keyBuf1[0] = personID;
    keyNum = 0;

    status = BTRVID(
        B_GET_EQUAL,
        posBlock1,
        &personRecord,
```

```

        &dataLen,
        keyBuf1,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

printf("Btrieve B_GET_EQUAL status = %d\n", status);
if (status == B_NO_ERROR)
{
    printf("\n");
    printf("The retrieved record is:\n");
    printf("ID:%ld\n", personRecord.ID);
    printf("Name:%s %s\n", personRecord.FirstName,
           personRecord.LastName );
    printf("Street:%s\n", personRecord.Street);
    printf("City:%s\n", personRecord.City);
    printf("State:%s\n", personRecord.State);
    printf("Zip:%s\n", personRecord.Zip);
    printf("Country:%s\n", personRecord.Country);
    printf("Phone:%s\n", personRecord.Phone);
    printf("\n");
}
}

/* Stat オペレーションを実行し、fileCreateBuf (作成バッファ) を取得 */
memset(&fileCreateBuf, 0, sizeof(fileCreateBuf));
dataLen = sizeof(fileCreateBuf);
keyNum = (BTI_WORD)-1;

status = BTRVID(B_STAT,
               posBlock1,
               &fileCreateBuf,
               &dataLen,
               keyBuf1,
               keyNum,
               (BTI_BUFFER_PTR)&clientID);

if (status == B_NO_ERROR)
{
    /* sample2.btr を作成して開く */
    keyNum = 0;
    dataLen = sizeof(fileCreateBuf);
    status = BTRVID(B_CREATE,
                   posBlock2,
                   &fileCreateBuf,
                   &dataLen,
                   keyBuf2,
                   keyNum,
                   (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_CREATE status = %d\n", status);
}

```

Btrieve API の言語インターフェイス

```
}

if (status == B_NO_ERROR)
{
    keyNum = 0;
    dataLen = 0;

    status = BTRVID(
        B_OPEN,
        posBlock2,
        dataBuf,
        &dataLen,
        keyBuf2,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_OPEN status (sample2.btr) = %d\n", status);
    if (status == B_NO_ERROR)
    {
        file2Open = TRUE;
    }
}

/* オリジナルのファイルからデータを抽出し、新しいファイルへ挿入 */
if (status == B_NO_ERROR)
{
    /* getFirst を実行してカレンシーを確立 */
    keyNum = 2; /* STATE-CITY index */
    memset(&personRecord, 0, sizeof(personRecord));
    memset(&keyBuf2[0], 0, sizeof(keyBuf2));
    dataLen = sizeof(personRecord);

    getStatus = BTRVID(
        B_GET_FIRST,
        posBlock1,
        &personRecord,
        &dataLen,
        keyBuf1,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_GET_FIRST status (sample.btr) = %d\n\n",
    getStatus);
}

gneBuffer = malloc(sizeof(GNE_BUFFER));
if (gneBuffer == NULL)
{
    printf("バッファに割り当てるメモリが不足しています");
    return(EXIT_WITH_ERROR);
}
```

```

}
memset(gneBuffer, 0, sizeof(GNE_BUFFER));
memcpy(&gneBuffer->preBuf.gneHeader.currencyConst[0], "UC", 2);
while (getStatus == B_NO_ERROR)
{
    gneBuffer->preBuf.gneHeader.rejectCount = 0;
    gneBuffer->preBuf.gneHeader.numberTerms = 2;
    posCtr = sizeof(GNE_HEADER);

    /* 最初の条件を代入 */
    gneBuffer->preBuf.term1.fieldType = 11;
    gneBuffer->preBuf.term1.fieldLen = 3;
    gneBuffer->preBuf.term1.fieldOffset = 108;
    gneBuffer->preBuf.term1.comparisonCode = 1;
    gneBuffer->preBuf.term1.connector = 2;
    memcpy(&gneBuffer->preBuf.term1.value[0], "TX", 2);
    posCtr += sizeof(TERM_HEADER);

    /* 2 つ目の条件を代入 */
    gneBuffer->preBuf.term2.fieldType = 11;
    gneBuffer->preBuf.term2.fieldLen = 3;
    gneBuffer->preBuf.term2.fieldOffset = 108;
    gneBuffer->preBuf.term2.comparisonCode = 1;
    gneBuffer->preBuf.term2.connector = 0;
    memcpy(&gneBuffer->preBuf.term2.value[0], "CA", 2);
    posCtr += sizeof(TERM_HEADER);

    /* プロジェクション ヘッダーを設定してレコード全体を読み込む */
    gneBuffer->preBuf.retrieval.maxRecsToRetrieve = 20;
    gneBuffer->preBuf.retrieval.noFieldsToRetrieve = 1;
    posCtr += sizeof(RETRIEVAL_HEADER);
    gneBuffer->preBuf.recordRet.fieldLen = sizeof(PERSON_STRUCT);
    gneBuffer->preBuf.recordRet.fieldOffset = 0;
    posCtr += sizeof(FIELD_RETRIEVAL_HEADER);
    gneBuffer->preBuf.gneHeader.descriptionLen = posCtr;

    dataLen = sizeof(GNE_BUFFER);
    getStatus = BTRVID(
        B_GET_NEXT_EXTENDED,
        posBlock1,
        gneBuffer,
        &dataLen,
        keyBuf1,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_GET_NEXT_EXTENDED status = %d\n", getStatus);
}

```

Btrieve API の言語インターフェイス

```
/* Get Next Extended はファイルの終わりに達してもレコードを返すことがある */
if ((getStatus == B_NO_ERROR) || (getStatus == B_END_OF_FILE))
{
    printf("GetNextExtended returned %d records.\n", gneBuffer-
>postBuf.numReturned);
    for (i = 0; i < gneBuffer->postBuf.numReturned; i++)
    {
        dataLen = sizeof(PERSON_STRUCT);
        memcpy(dataBuf, &gneBuffer->postBuf.recs[i].personRecord,
dataLen);
        status = BTRVID(
            B_INSERT,
            posBlock2,
            dataBuf,
            &dataLen,
            keyBuf2,
            -1, /* カレンシー変更なし */
            (BTI_BUFFER_PTR) &clientID);
    }
    printf("Inserted %d records in new file, status = %d\n\n",
gneBuffer->postBuf.numReturned, status);
}
memset(gneBuffer, 0, sizeof(GNE_BUFFER));
memcpy(&gneBuffer->preBuf.gneHeader.currencyConst[0], "EG", 2);
}

free(gneBuffer);
gneBuffer = NULL;

/* 開いているファイルを閉じる */
if (file1Open)
{
    dataLen = 0;
    status = BTRVID(
        B_CLOSE,
        posBlock1,
        dataBuf,
        &dataLen,
        keyBuf1,
        keyNum,
        (BTI_BUFFER_PTR) &clientID);
    printf("Btrieve B_CLOSE status (sample.btr) = %d\n", status);
}
if (file2Open)
{
    dataLen = 0;
    status = BTRVID(
        B_CLOSE,
        posBlock2,
        dataBuf,
```

```

        &dataLen,
        keyBuf2,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_CLOSE status (sample2.btr) = %d\n", status);
}

return(status);
}

```

Btrieve Login の C++ プログラムの例

loginapi.c と同様に Web ダウンロードにより 提供される 以下のプログラム例では、他のオペレーションのコンテキストで Btrieve のログイン操作を実行する方法について説明します。この操作は、MicroKernel との依存関係で要求される順番（ファイルを開いてから I/O を実行するなど）で行われます。

Btrieve オペレーションの例については、「[C++ プログラムの例](#)」を参照してください。

```

/*****
** Copyright 1982-2003 Pervasive Software Inc. All Rights Reserved
*****/
/*****
LOGINAPI.C

```

これは、目的の環境で、コンパイラ ツールを使用した Btrieve アプリケーションのコンパイル、リンク、実行が可能であることを確認するための簡単なサンプルです。

このプログラムは、セキュリティで保護されたデータベースを使用した Btrieve ログイン API の使用方法を示します。ログイン API は Pervasive.SQL 8.50 以降のバージョンでは使用されていません。プログラムを実行するには、DefaultDb データベースをセキュリティで保護し、セキュリティ モードにデータベース認証および許可を設定してください。これを設定するには PCC を使用します。

このプログラムでは、DefaultDb データベースで以下の操作を行います。

- Btrieve オペレーション B_LOGIN (78) を使用して DefaultDb データベースにログインする
- sample.btr を開く
- Key 0 の既知の値を持つレコードを取得する
- 読み込んだレコードを表示する
- sample.btr を閉じる
- DefaultDB データベースからログアウトする

次のパートでは Btrieve Open 呼び出しで URI を使用方法を示します。

- キー バッファで URI 文字列を使用して sample.btr を開く
- Key 0 の既知の値を持つレコードを取得する
- 読み込んだレコードを表示する
- sample.btr を閉じる

Btrieve API の言語インターフェイス

重要: Btrieve データ ファイルのサンプル `sample.btr` が保存されているディレクトリは、フルパスを指定してください。また、それぞれの「重要」項目も参考にしてください。

このプログラムは、インターフェイス モジュールが対応するプラットフォームで実行できます。プラットフォームは、'btrapi.h' でリストされるプラットフォームスイッチで表されます。Windows の場合は、コンソール アプリケーションで標準出力 `printf()` を使用できるようにしてください。
メモ: 多くの C/C++ コンパイラは、標準 I/O を使用した Windows アプリケーションに対応しています。

`btrapi.h` の冒頭部分で、アプリケーションのターゲット プラットフォーム選択に関する情報を参照してください。ターゲット プラットフォームを指定してください。

```
*****/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <btrapi.h>
#include <btrconst.h>

/*****
  定数
*****/
/*****
  重要: 以下を変更し、使用する sample.btr へのフルパスを指定してください。
*****/
#ifdef BTI_LINUX
#define FILE_NAME "/usr/local/psql/data/samples/sample.btr"
#else
#ifdef BTI_WIN_32
#define FILE_NAME "<path>%samples%sample.btr"
#endif
#endif

#define EXIT_WITH_ERROR 1
#define TRUE 1
#define FALSE 0
#define MY_THREAD_ID 50

/* 弊社の構造体の定義を加えないでください。*/
#ifdef __BORLANDC__
#pragma option -a-
#else
#ifdef __MSC_VER || defined(__WATCOMC__)
#pragma pack(1)
#endif
#endif
#endif
```

```

/*****
クライアント ID とバージョンの構造体の型定義
*****/
typedef struct
{
    BTI_CHAR networkAndNode[12];
    BTI_CHAR applicationID[2];
    BTI_WORD threadID;
} CLIENT_ID;

typedef struct
{
    BTI_SINT Version;
    BTI_SINT Revision;
    BTI_CHAR MKDEId;
} VERSION_STRUCT;

/*****
sample.btr のレコードの定義
*****/
typedef struct
{
    BTI_LONG ID;
    BTI_CHAR FirstName[16];
    BTI_CHAR LastName[26];
    BTI_CHAR Street[31];
    BTI_CHAR City[31];
    BTI_CHAR State[3];
    BTI_CHAR Zip[11];
    BTI_CHAR Country[21];
    BTI_CHAR Phone[14];
} PERSON_STRUCT;

/* 構造体パッキングの復元 */
#if defined(__BORLANDC__)
    #pragma option -a.
#else
    #if defined(_MSC_VER) || defined(__WATCOMC__)
        #pragma pack()
    #endif
#endif

/*****
メイン
*****/
int main(void)
{
    /* Btrieve 関数パラメーター */
    BTI_BYTE posBlock[128];
    BTI_BYTE dataBuf[255];

```

Btrieve API の言語インターフェイス

```
BTI_WORD  dataLen;
BTI_BYTE  keyBuf[255];
BTI_WORD  keyNum = 0;

BTI_LONG  personID;
BTI_BYTE  fileOpen = FALSE;
BTI_SINT  status = 0;

CLIENT_ID  clientID;
PERSON_STRUCT  personRecord;

/* URI 文字列を定義して DefaultDb データベースの sample.btr を開く */
/* DefaultDb データベースに対し、ユーザー名を "Master"、パスワードを */
/* "master" で SQL セキュリティを設定 */

BTI_CHAR *URI_STRING1 =
"btrv://Master@localhost/defaultdb?pwd=master";
BTI_CHAR *URI_STRING2 =
"btrv://Master@localhost/defaultdb?file=/usr/local/psql/data/samples/s
ample.btr&pwd=master";

printf("***** Btrieve C/C++ Interface Demo for Login API *****\n\n");

/* クライアント ID のセットアップ */
memset(clientID.networkAndNode, 0, sizeof(clientID.networkAndNode));
memcpy(clientID.applicationID, "MT", 2); /* "AA" > 以上を設定 */
clientID.threadID = MY_THREAD_ID;

/* セキュリティで保護されたデータベースにログイン */
memset(dataBuf, 0, sizeof(dataBuf));
strcpy(keyBuf, URI_STRING1);
keyNum = 0;
status = BTRVID(
    B_LOGIN,
    posBlock,
    dataBuf,
    &dataLen,
    keyBuf,
    keyNum,
    (BTI_BUFFER_PTR)&clientID);

printf("Btrieve B_LOGIN status = %d\n", status);

/* バッファのクリア */
memset(dataBuf, 0, sizeof(dataBuf));
memset(keyBuf, 0, sizeof(keyBuf));

/* sample.btr を開く */
strcpy((BTI_CHAR *)keyBuf, FILE_NAME);
```

```

keyNum = 0;
dataLen = 0;

status = BTRVID(
    B_OPEN,
    posBlock,
    dataBuf,
    &dataLen,
    keyBuf,
    keyNum,
    (BTI_BUFFER_PTR)&clientID);

printf("Btrieve B_OPEN status (sample.btr) = %d\n", status);
if (status == B_NO_ERROR)
{
    fileOpen = TRUE;
}

/* B_GET_EQUAL を使用して key 0 = 263512477 のレコードを取得 */
if (status == B_NO_ERROR)
{
    memset(&personRecord, 0, sizeof(personRecord));
    dataLen = sizeof(personRecord);
    personID = 263512477; /* 実際これは社会保障番号 */
    *(BTI_LONG BTI_FAR *)&keyBuf[0] = personID;
    keyNum = 0;

    status = BTRVID(
        B_GET_EQUAL,
        posBlock,
        &personRecord,
        &dataLen,
        keyBuf,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_GET_EQUAL status = %d\n", status);
    if (status == B_NO_ERROR)
    {
        printf("\n");
        printf("The retrieved record is:\n");
        printf("ID:%ld\n", personRecord.ID);
        printf("Name:%s %s\n", personRecord.FirstName,
            personRecord.LastName );
        printf("Street:%s\n", personRecord.Street);
        printf("City:%s\n", personRecord.City);
        printf("State:%s\n", personRecord.State);
        printf("Zip:%s\n", personRecord.Zip);
    }
}

```

Btrieve API の言語インターフェイス

```
        printf("Country:%s\n", personRecord.Country);
        printf("Phone:%s\n", personRecord.Phone);
        printf("%n");
    }
}

/* 開いているファイルを閉じる */
if (fileOpen)
{
    dataLen = 0;

    status = BTRVID(
        B_CLOSE,
        posBlock,
        dataBuf,
        &dataLen,
        keyBuf,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_CLOSE status (sample.btr) = %d\n", status);
}

/* セキュリティで保護されたデータベースからログアウト */

memset(dataBuf, 0, sizeof(dataBuf));
strcpy(keyBuf, URI_STRING1);
keyNum = 1;
status = BTRVID(
    B_LOGIN,
    posBlock,
    dataBuf,
    &dataLen,
    keyBuf,
    keyNum,
    (BTI_BUFFER_PTR)&clientID);

printf("Btrieve logout status = %d\n", status);
printf("%n\n");

/* 以下の例では Btrieve Open 呼び出しで URI 文字列を使用する */

printf("*** 以下の例では Btrieve Open 呼び出しで URI 文字列を使用する
***\n\n");

/* クライアント ID のセットアップ */
memset(clientID.networkAndNode, 0, sizeof(clientID.networkAndNode));
```

```

memcpy(clientID.applicationID, "MT", 2); /* "AA" > 以上を設定 */
clientID.threadID = MY_THREAD_ID;

/* バッファのクリア */
memset(dataBuf, 0, sizeof(dataBuf));
memset(keyBuf, 0, sizeof(keyBuf));

/* sample.btr を開く */
strcpy((BTI_CHAR *)keyBuf, URI_STRING2);

keyNum = 0;
dataLen = 0;

status = BTRVID(
    B_OPEN,
    posBlock,
    dataBuf,
    &dataLen,
    keyBuf,
    keyNum,
    (BTI_BUFFER_PTR)&clientID);

printf("Btrieve B_OPEN status (sample.btr) = %d\n", status);
if (status == B_NO_ERROR)
{
    fileOpen = TRUE;
}

/* B_GET_EQUAL を使用して key 0 = 263512477 のレコードを取得 */
if (status == B_NO_ERROR)
{
    memset(&personRecord, 0, sizeof(personRecord));
    dataLen = sizeof(personRecord);
    personID = 263512477; /* 実際これは社会保障番号 */
    *(BTI_LONG BTI_FAR *)&keyBuf[0] = personID;
    keyNum = 0;

    status = BTRVID(
        B_GET_EQUAL,
        posBlock,
        &personRecord,
        &dataLen,
        keyBuf,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_GET_EQUAL status = %d\n", status);
    if (status == B_NO_ERROR)
    {
        printf("%n");
    }
}

```

Btrieve API の言語インターフェイス

```
printf("The retrieved record is:\n");
printf("ID:%ld\n", personRecord.ID);
printf("Name:%s %s\n", personRecord.FirstName,
        personRecord.LastName );
printf("Street:%s\n", personRecord.Street);
printf("City:%s\n", personRecord.City);
printf("State:%s\n", personRecord.State);
printf("Zip:%s\n", personRecord.Zip);
printf("Country:%s\n", personRecord.Country);
printf("Phone:%s\n", personRecord.Phone);
printf("\n");
}
}

/* 開いているファイルを閉じる */
if (fileOpen)
{
    dataLen = 0;

    status = BTRVID(
        B_CLOSE,
        posBlock,
        dataBuf,
        &dataLen,
        keyBuf,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    printf("Btrieve B_CLOSE status (sample.btr) = %d\n", status);
}

return(status);
}
```

サンプルプログラムのコンパイル、リンク、実行

Btrieve 用 C インターフェイスは、プラットフォームとコンパイラにより操作が異なります。通常は、以下の操作を行います。

- BtrSamp.c ファイルで、Sample.btr および Sample2.btr のパスを適切なものに変更します。
- 適切なインポート ライブラリを確認します。場所は以下のとおりです。

Microsoft	¥Intf¥ImpLib¥W32
Watcom および Borland	¥Intf¥ImpLib¥W32x

- プラットフォーム スイッチを設定します。スイッチについては、『Pervasive PSQL Programmer's Guide』を参照してください。
- ディレクトリ設定を更新します。現在のディレクトリを含めます。
- 適切な構造体アライメントを使用します。サンプル プログラムには、構造体アライメント用のプラグマが含まれています。実際のアプリケーションでも同様のプラグマを使用、またはコンパイラ スイッチを構造体アライメント用に設定する必要があるため、このセクションでは、使用するコンパイラ スイッチを明確にします。

このセクションでは、一般的な開発環境に共通する操作方法を説明します。お使いの開発環境が該当しない場合は、この操作方法を参考にしてください。

➤ Microsoft Visual C++ では、サンプル プログラムのコンパイル、リンク、実行を、以下の方法で行います。

- 1 Developer Studio プログラミング環境では、[ファイル] メニューの [新規作成] を選択します。
- 2 [新規作成] ダイアログの [プロジェクト] タブで以下の操作を行います。
 - a. プロジェクトの種類を "Win32 Console Application" に設定します (BtrSamp は DOS プロンプトから実行)。
 - b. [プロジェクト名] に "BtrSamp" を設定します。
 - c. [位置] に %Intf%C ディレクトリへの場所を設定します。
 - d. [OK] をクリックします。

BtrSamp という新しいプロジェクトが作成されます。

- 3 ワークスペースの [FileView] タブで BtrSamp プロジェクトを右クリックし、[ファイルをプロジェクトへ追加] コマンドを選択して以下のファイルを追加します。
 - %Intf%C%BtrApi.c
 - %Intf%C%BtrSamp.c
 - %Intf%ImpLib%Win32%W3Btrv7.lib (Windows 32 ビットの場合)
 - %Intf%ImpLib%Win64%W6Btrv7.lib (Windows 64 ビットの場合)
 - [OK] をクリックします。
- 4 BtrSamp.c ファイルで、Sample.btr および Sample2.btr のパスを適切なものに変更します。
- 5 [プロジェクト] メニューの [設定] をクリックします。[C++] タブで以下の操作を行います。
 - "プリプロセッサ" カテゴリで、[プリプロセッサの定義] ボックスに BTI_WIN_32 を追加し、[インクルード ファイルのパス] ボックスに現在のディレクトリ (.) を追加します。

- "コード生成" カテゴリでは、[構造体メンバーのアライメント]を "1 バイト" に設定します。
 - [OK] をクリックします。
- 6 [ビルド] メニューの [ビルド] をクリックします。
サンプル プログラムが作成され、プロジェクト ファイル ディレクトリに BtrSamp.exe が保存されます。
- 7 DOS プロンプト ウィンドウから BtrSamp.exe を実行します。

➤ Watcom C++ では、サンプル プログラムのコンパイル、リンク、実行を、以下の方法で行います。

- 1 IDE プログラミング環境では、[File] メニューの [New Project] を選択します。
- 2 [Enter Project Filename] ダイアログで、File Name を "BtrSamp.wpj" に設定して [OK] をクリックします。
- 3 [New Target] ダイアログで以下の操作を行います。
 - [ターゲット名] を "BtrSamp" に設定します。
 - Target Environment を "Win32" に設定します。
 - Image Type を "Executable [.exe]" に設定します。
 - [OK] をクリックします。
- 4 [Sources] メニューの [New Source] を選択し、以下のファイルを追加します。
 - ¥Intf¥C¥BtrApi.c
 - ¥Intf¥C¥BtrSamp.c
 - ¥Intf¥ImpLib¥Win32x¥W3Btrv7.lib
 - [OK] をクリックします。
- 5 BtrSamp.c ファイルで、Sample.btr および Sample2.btr のパスを適切なものに変更します。
- 6 [Sources] メニューの [Source Options] サブメニューから [C Compiler Switches] を選択し、以下の操作を行います。
 - 以下の例のように、[File Option Switches] ダイアログで現在のディレクトリを [Include Directories] ボックスに追加します。
\$(%watcom)¥h;\$(%watcom)¥h¥nt; .
 - [Source Switches] ダイアログで、[Macro Definitions] ボックスに BTI_WIN_32 を追加し、Structure Alignment を "Pack structures" に設定します。
 - [OK] をクリックします。

- 7 ツールバーの [Make All Targets] ボタンをクリックします。
サンプル プログラムが作成され、プロジェクト ファイル ディレクトリに BtrSamp.exe が保存されます。
 - 8 DOS プロンプト ウィンドウから BtrSamp.exe を実行します。
- Borland C++ では、サンプル プログラムのコンパイル、リンク、実行を、以下の方法で行います。
- 1 Borland C++ プログラミング環境では、[プロジェクト] メニューの [新規作成] を選択します。
 - 2 [新規ターゲット] ダイアログで以下の操作を行います。
 - [プロジェクト パス名] を "¥Intf¥C¥BtrSamp" に設定します。
 - [ターゲット名] を "BtrSamp" に設定します。
 - [ターゲットの種類] を "アプリケーション[.exe]" に設定します。
 - [ターゲット モデル] を "Console" に設定します。
 - [フレームワーク] グループの [クラス ライブラリ] チェック ボックスをオフにします。
 - [拡張設定] ボタンをクリックし、[初期ノード] グループで ".C ノード" を指定して [.rc] チェック ボックスと [.def] チェック ボックスをオフにします。
 - [OK] をクリックします。
 - 3 [プロジェクト] ウィンドウで BtrSamp プロジェクトを右クリックし、[ノード追加] を選択して以下のファイルを追加します。
 - ¥Intf¥C¥BtrApi.c
 - ¥Intf¥C¥BtrSamp.c
 - ¥Intf¥ImpLib¥Win32x¥W3Btrv7.lib
 - [OK] をクリックします。
 - 4 BtrSamp.c ファイルで、Sample.btr および Sample2.btr のパスを適切なものに変更します。
 - 5 [オプション] メニューの [プロジェクト] をクリックし、以下の操作を行います。
 - 以下の例のように、"ディレクトリ" トピックを選択し、[ソース ディレクトリ] グループの [インクルード] リストに現在のディレクトリを追加します。

```
c:¥bc5¥include;.
```
 - "コンパイラ" で "マクロ定義" トピックを選択し、[マクロ定義] ボックスに BTI_WIN_32 を追加します。
 - "32ビットコンパイラ" で "プロセッサ" トピックを選択し、[データの配置] を "Byte" に設定します。
 - [OK] をクリックします。

- 6 [プロジェクト] メニューの [再構築] を選択します。
サンプル プログラムが作成され、プロジェクト ファイル ディレクトリに BtrSamp.exe が保存されます。
- 7 DOS プロンプト ウィンドウから BtrSamp.exe を実行します。

C++ Builder アプリケーションのコンパイル

➤C++ Builder では、サンプル プログラムのコンパイル、リンク、実行を、以下の方法で行います。

- 1 [ファイル] メニューの [プロジェクトを開く] から CBBtrv.mak ファイルを選択します。
- 2 [プロジェクト] メニューの [オプション] を選択し、[ディレクトリ / 条件] タブから以下の操作を行います。
 - [インクルード パス] にディレクトリ "%Intf\C" を追加します。
 - [ライブラリ パス] にディレクトリ "%Intf\Implib\Win32x" を追加します。
- 3 [プロジェクト] メニューの [メイク] を選択します。
サンプル プログラムが作成され、プロジェクト ファイル ディレクトリに CBBtrv.exe が保存されます。
- 4 [Run] メニューの [Run] を選択します。
- 5 [Btrieve Sample Application] ウィンドウの [Run Test] ボタンをクリックします。

以下の C++ Builder プログラムの例では、Btrieve の一般的な操作の実行方法をいくつか説明します。これらの操作は、MicroKernel との依存関係で要求される順番 (ファイルを開いてから I/O を実行するなど) で行われます。

CBBMain.cpp

```
/*
*****
**
** Copyright 2003 Pervasive Software Inc.
** All Rights Reserved
*****
/
*****
/
CBBMain.cpp
これは、目的の環境で、コンパイラ ツールを使用した Btrieve アプリケーションの
コンパイル、リンク、実行が可能なことを確認するための簡単なサンプルです。

このプログラムでは、Btrieve の C/C++ インターフェイスを示します。
これは、Borland C++ Builder がインストールされた MS Windows NT および
Windows 9X/ME 環境での例です。
```

このプログラムでは、サンプル データベースで以下の操作を行います。

- Microkernel Database エンジンのバージョン情報を取得する
- sample.btr を開く
- Key 0 の既知の値を持つレコードを取得する
- 読み込んだレコードを表示する
- Stat オペレーションを実行する
- 空の sample.btr のクローンを作成して開く
- Get Next Extended オペレーションを実行して sample.btr のレコードのサブセットを抽出する
- クローン ファイルにこれらのレコードを挿入する
- 両ファイルを閉じる

このサンプルをコンパイルするには、IDE で使用可能な以下のファイルを、Btrieve "c" インターフェイス ディレクトリから作成する必要があります。

- btrapi.c
- btrapi.h
- btrconst.h

重要: Btrieve データ ファイルのサンプル sample.btr が保存されているディレクトリへのフルパスを必ず指定する必要があります。また、下の「重要」項目も参考してください。

```

*****/
//-----
#include <vcl\vcl.h>
#pragma hdrstop

#include "CBBMain.h"
#include <btrapi.h>
#include <btrconst.h>
//-----
#pragma resource "*.dfm"

/*****
定数
*****/
#define EXIT_WITH_ERROR      1
#define TRUE                 1
#define FALSE                0
#define VERSION_OFFSET      0
#define REVISION_OFFSET     2
#define PLATFORM_ID_OFFSET  4
#define MY_THREAD_ID        50

/* 弊社の構造体に定義を加えないでください。*/
/* Borland のヘルプではこれがデフォルトとなっていますが、正しくありません。*/
#pragma option -a1

```

Btrieve API の言語インターフェイス

```

/*****
クライアント ID とバージョンの構造体の型定義
*****/
typedef struct
{
    BTI_CHAR    networkAndNode[12];
    BTI_CHAR    applicationID[2];
    BTI_WORD    threadID;
} CLIENT_ID;

typedef struct
{
    BTI_SINT    Version;
    BTI_SINT    Revision;
    BTI_CHAR    MKDEId;
} VERSION_STRUCT;

/*****
sample.btr からのレコードの定義
*****/
typedef struct
{
    BTI_LONG    ID;
    BTI_CHAR    FirstName[16];
    BTI_CHAR    LastName[26];
    BTI_CHAR    Street[31];
    BTI_CHAR    City[31];
    BTI_CHAR    State[3];
    BTI_CHAR    Zip[11];
    BTI_CHAR    Country[21];
    BTI_CHAR    Phone[14];
} PERSON_STRUCT;

/*****
Stat/Create 構造体の型定義
*****/
typedef struct
{
    BTI_SINT    recLength;
    BTI_SINT    pageSize;
    BTI_SINT    indexCount;
    BTI_CHAR    reserved[4];
    BTI_SINT    flags;
    BTI_BYTE    dupPointers;
    BTI_BYTE    notUsed;
    BTI_SINT    allocations;
} FILE_SPECS;

typedef struct
{

```

```

BTI_SINT  position;
BTI_SINT  length;
BTI_SINT  flags;
BTI_CHAR  reserved[4];
BTI_CHAR  type;
BTI_CHAR  null;
BTI_CHAR  notUsed[2];
BTI_BYTE  manualKeyNumber;
BTI_BYTE  acsNumber;
} KEY_SPECS;

typedef struct
{
    FILE_SPECS  fileSpecs;
    KEY_SPECS   keySpecs[5];
} FILE_CREATE_BUF;

/*****
    Get Next Extended 処理用の構造体の型定義
*****/
typedef struct
{
    BTI_SINT    descriptionLen;
    BTI_CHAR    currencyConst[2];
    BTI_SINT    rejectCount;
    BTI_SINT    numberTerms;
} GNE_HEADER;

typedef struct
{
    BTI_CHAR    fieldType;
    BTI_SINT    fieldLen;
    BTI_SINT    fieldOffset;
    BTI_CHAR    comparisonCode;
    BTI_CHAR    connector;
    BTI_CHAR    value[3];
} TERM_HEADER;

typedef struct
{
    BTI_SINT    maxRecsToRetrieve;
    BTI_SINT    noFieldsToRetrieve;
} RETRIEVAL_HEADER;

typedef struct
{
    BTI_SINT    fieldLen;
    BTI_SINT    fieldOffset;
} FIELD_RETRIEVAL_HEADER;

```

Btrieve API の言語インターフェイス

```
typedef struct
{
    GNE_HEADER          gneHeader;
    TERM_HEADER        term1;
    TERM_HEADER        term2;
    RETRIEVAL_HEADER    retrieval;
    FIELD_RETRIEVAL_HEADER recordRet;
} PRE_GNE_BUFFER;

typedef struct
{
    BTI_SINT          recLen;
    BTI_LONG          recPos;
    PERSON_STRUCT    personRecord;
} RETURNED_REC;

typedef struct
{
    BTI_SINT          numReturned;
    RETURNED_REC     recs[20];
} POST_GNE_BUFFER;

typedef union
{
    PRE_GNE_BUFFER   preBuf;
    POST_GNE_BUFFER  postBuf;
} GNE_BUFFER, BTI_FAR* GNE_BUFFER_PTR;

//-----
// エクスポートされない事前宣言
void printLB(TListBox *LB, char *msg);

TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
void __fastcall TForm1::ExitButtonClick(TObject *Sender)
{
    Form1->Close();
}
//-----
void __fastcall TForm1::RunButtonClick(TObject *Sender)
{
    runTest();
}
```

```

/*****
    ListBox に書き出すヘルパー関数
    *****/
void printLB(TListBox *LB, char *msg)
{
    LB->Items->Add(msg);
}

/*****
    ここですべての処理が行われる
    *****/
BTI_SINT runTest(void)
{
    /* Btrieve 関数パラメーター */
    BTI_BYTE posBlock1[128];
    BTI_BYTE posBlock2[128];
    BTI_BYTE dataBuf[255];
    BTI_WORD dataLen;
    BTI_BYTE keyBuf1[255];
    BTI_BYTE keyBuf2[255];
    BTI_WORD keyNum = 0;

    BTI_BYTE btrieveLoaded = FALSE;
    BTI_LONG personID;
    BTI_BYTE file1Open = FALSE;
    BTI_BYTE file2Open = FALSE;
    BTI_SINT status;
    BTI_SINT getStatus;
    BTI_SINT i;
    BTI_SINT posCtr;

    CLIENT_ID    clientID;
    VERSION_STRUCT versionBuffer[3];
    FILE_CREATE_BUF fileCreateBuf;
    GNE_BUFFER_PTR gneBuffer;
    PERSON_STRUCT personRecord;

    BTI_CHAR tmpBuf[1024];

    /*****
        プログラム タイトルの出力
        *****/
    printLB(Form1->ListBox1,
        "**** Btrieve -- C++ Builder Sample ****" );

    /* クライアント ID のセットアップ */
    memset(clientID.networkAndNode, 0,
        sizeof(clientID.networkAndNode));
    memcpy(clientID.applicationID, "MT", 2); /* "AA" 以上を設定 */
    clientID.threadID = MY_THREAD_ID;

```

Btrieve API の言語インターフェイス

```
memset(versionBuffer, 0, sizeof(versionBuffer));
dataLen = sizeof(versionBuffer);

status = BTRVID(
    B_VERSION,
    posBlock1,
    &versionBuffer,
    &dataLen,
    keyBuf1,
    keyNum,
    (BTI_BUFFER_PTR)&clientID);

if (status == B_NO_ERROR)
{
    strcpy(tmpBuf, "Btrieve Versions returned are: ");
    printLB(Form1->ListBox1, tmpBuf);
    for (i = 0; i < 3; i++) {
        if (versionBuffer[i].Version != 0)
        {
            sprintf(tmpBuf, "    %d.%d %c", versionBuffer[i].Version,
                versionBuffer[i].Revision,
                versionBuffer[i].MKDEId );
            printLB(Form1->ListBox1, tmpBuf);
        }
    }
    printLB(Form1->ListBox1, "");
    btrieveLoaded = TRUE;
}
else
{
    sprintf(tmpBuf, "Btrieve B_VERSION status = %d", status );
    printLB(Form1->ListBox1, tmpBuf);
    if (status != B_RECORD_MANAGER_INACTIVE)
    {
        btrieveLoaded = TRUE;
    }
}

/* バッファークリア */
if (status == B_NO_ERROR)
{
    memset(dataBuf, 0, sizeof(dataBuf));
    memset(keyBuf1, 0, sizeof(keyBuf1));
    memset(keyBuf2, 0, sizeof(keyBuf2));
}

/* sample.btr を開く */
if (status == B_NO_ERROR)
{
```

```

/*****
  重要：以下を変更し、使用する sample.btr へのフルパスを指定してください。
*****/
strcpy( (BTI_CHAR *)keyBuf1, "c:\\sample\\sample.btr");
strcpy( (BTI_CHAR *)keyBuf2, "c:\\sample\\sample2.btr");

keyNum = 0;
dataLen = 0;

status = BTRVID(
    B_OPEN,
    posBlock1,
    dataBuf,
    &dataLen,
    keyBuf1,
    keyNum,
    (BTI_BUFFER_PTR)&clientID);

sprintf(tmpBuf, "Btrieve B_OPEN status = %d", status );
printLB(Form1->ListBox1, tmpBuf);
if (status == B_NO_ERROR)
{
    file1Open = TRUE;
}
}

/* B_GET_EQUAL を使用して key 0 = 263512477 のレコードを取得 */
if (status == B_NO_ERROR)
{
    memset(&personRecord, 0, sizeof(personRecord));
    dataLen = sizeof(personRecord);
    personID = 263512477; /* 実際これは社会保障番号 */
    *(BTI_LONG BTI_FAR *)&keyBuf1[0] = personID;
    keyNum = 0;

    status = BTRVID(
        B_GET_EQUAL,
        posBlock1,
        &personRecord,
        &dataLen,
        keyBuf1,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    sprintf(tmpBuf, "Btrieve B_GET_EQUAL status = %d", status );
    printLB(Form1->ListBox1, tmpBuf);
    if (status == B_NO_ERROR)
    {
        sprintf(tmpBuf, " ");
        printLB(Form1->ListBox1, tmpBuf);
    }
}

```

Btrieve API の言語インターフェイス

```
    sprintf(tmpBuf, "The retrieved record is:" );
    printLB(Form1->ListBox1, tmpBuf);
    sprintf(tmpBuf, "ID:%ld", personRecord.ID );
    printLB(Form1->ListBox1, tmpBuf);
    sprintf(tmpBuf, "Name:%s %s", personRecord.FirstName,
            personRecord.LastName );
    printLB(Form1->ListBox1, tmpBuf);
    sprintf(tmpBuf, "Street:%s", personRecord.Street );
    printLB(Form1->ListBox1, tmpBuf);
    sprintf(tmpBuf, "City:%s", personRecord.City );
    printLB(Form1->ListBox1, tmpBuf);
    sprintf(tmpBuf, "State:%s", personRecord.State );
    printLB(Form1->ListBox1, tmpBuf);
    sprintf(tmpBuf, "Zip:%s", personRecord.Zip );
    printLB(Form1->ListBox1, tmpBuf);
    sprintf(tmpBuf, "Country:%s", personRecord.Country );
    printLB(Form1->ListBox1, tmpBuf);
    sprintf(tmpBuf, "Phone:%s", personRecord.Phone );
    printLB(Form1->ListBox1, tmpBuf);
    sprintf(tmpBuf, "" );
    printLB(Form1->ListBox1, tmpBuf);
}
}

/* Stat オペレーションを実行し、fileCreateBuf (作成バッファ) を取得 */

memset(&fileCreateBuf, 0, sizeof(fileCreateBuf));
dataLen = sizeof(fileCreateBuf);
keyNum = (BTI_WORD)-1;

status = BTRVID(B_STAT,
                posBlock1,
                &fileCreateBuf,
                &dataLen,
                keyBuf1,
                keyNum,
                (BTI_BUFFER_PTR)&clientID);

if (status == B_NO_ERROR)
{
    /* sample2.btr を作成して開く */
    keyNum = 0;
    dataLen = sizeof(fileCreateBuf);
    status = BTRVID(B_CREATE,
                    posBlock2,
                    &fileCreateBuf,
                    &dataLen,
                    keyBuf2,
                    keyNum,
                    (BTI_BUFFER_PTR)&clientID);
}
```

```

    sprintf(tmpBuf, "Btrieve B_CREATE status = %d", status );
    printLB(Form1->ListBox1, tmpBuf);
}

if (status == B_NO_ERROR)
{
    keyNum = 0;
    dataLen = 0;

    status = BTRVID(
        B_OPEN,
        posBlock2,
        dataBuf,
        &dataLen,
        keyBuf2,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    sprintf(tmpBuf, "Btrieve B_OPEN status = %d", status );
    printLB(Form1->ListBox1, tmpBuf);
    if (status == B_NO_ERROR)
    {
        file2Open = TRUE;
    }
}

/* 元のファイルからデータを抽出し、新しいファイルへ挿入 */
if (status == B_NO_ERROR)
{
    /* getFirst を実行してカレンシーを確立 */
    keyNum = 2; /* STATE-CITY index */
    memset(&personRecord, 0, sizeof(personRecord));
    memset(&keyBuf2[0], 0, sizeof(keyBuf2));
    dataLen = sizeof(personRecord);

    getStatus = BTRVID(
        B_GET_FIRST,
        posBlock1,
        &personRecord,
        &dataLen,
        keyBuf1,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);

    sprintf(tmpBuf, "Btrieve B_GET_FIRST status = %d", getStatus );
    printLB(Form1->ListBox1, tmpBuf);
}

gneBuffer = (GNE_BUFFER_PTR)malloc(sizeof(GNE_BUFFER));

```

Btrieve API の言語インターフェイス

```
if (gneBuffer == NULL)
{
    strcpy(tmpBuf, "バッファーに割り当てるメモリが不足しています");
    printLB(Form1->ListBox1, tmpBuf);
    return(EXIT_WITH_ERROR);
}
memset(gneBuffer, 0, sizeof(GNE_BUFFER));
memcpy(&gneBuffer->preBuf.gneHeader.currencyConst[0], "UC", 2);
while (getStatus == B_NO_ERROR)
{
    gneBuffer->preBuf.gneHeader.rejectCount = 0;
    gneBuffer->preBuf.gneHeader.numberTerms = 2;
    posCtr = sizeof(GNE_HEADER);

    /* 最初の条件を代入 */
    gneBuffer->preBuf.term1.fieldType = 11;
    gneBuffer->preBuf.term1.fieldLen = 3;
    gneBuffer->preBuf.term1.fieldOffset = 108;
    gneBuffer->preBuf.term1.comparisonCode = 1;
    gneBuffer->preBuf.term1.connector = 2;
    memcpy(&gneBuffer->preBuf.term1.value[0], "TX", 2);
    posCtr += sizeof(TERM_HEADER);

    /* 2 つ目の条件を代入 */
    gneBuffer->preBuf.term2.fieldType = 11;
    gneBuffer->preBuf.term2.fieldLen = 3;
    gneBuffer->preBuf.term2.fieldOffset = 108;
    gneBuffer->preBuf.term2.comparisonCode = 1;
    gneBuffer->preBuf.term2.connector = 0;
    memcpy(&gneBuffer->preBuf.term2.value[0], "CA", 2);
    posCtr += sizeof(TERM_HEADER);

    /* プロジェクション ヘッダーを設定してレコード全体を読み込む */
    gneBuffer->preBuf.retrieval.maxRecsToRetrieve = 20;
    gneBuffer->preBuf.retrieval.noFieldsToRetrieve = 1;
    posCtr += sizeof(RETRIEVAL_HEADER);
    gneBuffer->preBuf.recordRet.fieldLen = sizeof(PERSON_STRUCT);
    gneBuffer->preBuf.recordRet.fieldOffset = 0;
    posCtr += sizeof(FIELD_RETRIEVAL_HEADER);
    gneBuffer->preBuf.gneHeader.descriptionLen = posCtr;

    dataLen = sizeof(GNE_BUFFER);
    getStatus = BTRVID(
        B_GET_NEXT_EXTENDED,
        posBlock1,
        gneBuffer,
        &dataLen,
        keyBuf1,
        keyNum,
        (BTI_BUFFER_PTR) &clientID);
}
```

```

    sprintf(tmpBuf, "Btrieve B_GET_NEXT_EXTENDED status = %d",
            getStatus );
    printLB(Form1->ListBox1, tmpBuf);

/* Get Next Extended は、ファイルの終わりに達しても */
/* レコードを返すことがあります */
    if ((getStatus == B_NO_ERROR) || (getStatus == B_END_OF_FILE))
    {
        sprintf(tmpBuf, "GetNextExtended returned %d records.",
                gneBuffer->postBuf.numReturned);
        printLB(Form1->ListBox1, tmpBuf);
        for (i = 0; i < gneBuffer->postBuf.numReturned; i++)
        {
            dataLen = sizeof(PERSON_STRUCT);
            memcpy(dataBuf, &gneBuffer->
                postBuf.recs[i].personRecord, dataLen);
            status = BTRVID(
                B_INSERT,
                posBlock2,
                dataBuf,
                &dataLen,
                keyBuf2,
                -1, /* カレンシー変更なし */
                (BTI_BUFFER_PTR)&clientID);
        }
        sprintf(tmpBuf, "Inserted %d records in new file, status = %d",
                gneBuffer->postBuf.numReturned, status);
        printLB(Form1->ListBox1, tmpBuf);
    }
    memset(gneBuffer, 0, sizeof(GNE_BUFFER));
    memcpy(&gneBuffer->preBuf.gneHeader.currencyConst[0], "EG", 2);
}

free(gneBuffer);
gneBuffer = NULL;

/* 開いているファイルを閉じる */
if (file1Open)
{
    dataLen = 0;

    status = BTRVID(
        B_CLOSE,
        posBlock1,
        dataBuf,
        &dataLen,
        keyBuf1,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);
}

```

Btrieve API の言語インターフェイス

```
    sprintf(tmpBuf, "Btrieve B_CLOSE status (sample.btr) = %d",
            status );
    printLB(Form1->ListBox1, tmpBuf);
}

if (file2Open)
{
    dataLen = 0;

    status = BTRVID(
        B_CLOSE,
        posBlock2,
        dataBuf,
        &dataLen,
        keyBuf2,
        keyNum,
        (BTI_BUFFER_PTR)&clientID);
    sprintf(tmpBuf, "Btrieve B_CLOSE status (sample2.btr) = %d",
            status );
    printLB(Form1->ListBox1, tmpBuf);
}
return(status);
}
```

COBOL

Web ダウンロードにより提供される以下のプログラム例では、Btrieve の一般的な操作方法を説明します。これらの操作は、MicroKernel との依存関係で要求される順番（ファイルを開いてから I/O を実行するなど）で行われます。

Btrsamp.cbl

```

*
*
* Copyright 2003 Pervasive Software Inc. All Rights Reserved
*
*
* BTRSAMP.CBL
* この COBOL プログラムのサンプルでは、Micro Focus COBOL v3.x を使用して
* アプリケーションから Btrieve 呼び出しを作成します。
*
*
IDENTIFICATION DIVISION.
*
PROGRAM-ID. TEST1.
*
*
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-PC.
OBJECT-COMPUTER. IBM-PC.
*
*
DATA DIVISION.
*
WORKING-STORAGE SECTION.
*
* BTRIEVE OP CODES
*
01 B-OPEN PIC 9(4) COMP-5 VALUE 0.
01 B-INSERT PIC 9(4) COMP-5 VALUE 2.
01 B-GETFIRST PIC 9(4) COMP-5 VALUE 12.
01 B-UPDATE PIC 9(4) COMP-5 VALUE 3.
01 B-CLOSE PIC 9(4) COMP-5 VALUE 1.
*
01 B-STATUS PIC 9(4) COMP-5 VALUE 0.
01 KEY-NUMBER PIC 9(4) COMP-5 VALUE 0.
01 BUF-LEN PIC 9(4) COMP-5 VALUE 0.
01 FILE-NAME PIC X(13) VALUE SPACES.
01 POSITION-BLOCK PIC X(128) VALUE SPACES.
01 DATA-BUFFER.

```

Btrieve API の言語インターフェイス

```
02 DECIMAL-FIELD          PIC 9(4) COMP-3 VALUE 0.
02 STRING-FIELD           PIC X(36) VALUE SPACES.
01 DSP-STATUS             PIC 9(5) COMP-5.
*
*
*
PROCEDURE DIVISION.
BEGIN.
*
* TEST.BTR を開く
*
*
MOVE 0    TO BUF-LEN.
MOVE 0    TO KEY-NUMBER.
MOVE 'TEST.BTR' TO FILE-NAME.
CALL "_BTRV" USING B-OPEN, B-STATUS, POSITION-BLOCK,
                DATA-BUFFER, BUF-LEN, FILE-NAME, KEY-NUMBER.
IF B-STATUS NOT = 0
    DISPLAY 'Error opening file.Status= ' B-STATUS
ELSE
    DISPLAY 'File ' FILE-NAME ' successfully opened'
END-IF.
*
* TEST.BTR への挿入
*
*
MOVE 1    TO DECIMAL-FIELD.
MOVE 'Record 1' TO STRING-FIELD.
MOVE 40   TO BUF-LEN.
MOVE 0    TO KEY-NUMBER
CALL "_BTRV" USING B-INSERT, B-STATUS, POSITION-BLOCK,
                DATA-BUFFER, BUF-LEN, FILE-NAME, KEY-NUMBER.
IF B-STATUS NOT = 0
    DISPLAY 'Error inserting into file.Status= ' B-STATUS
ELSE
    DISPLAY 'inserted:' DECIMAL-FIELD STRING-FIELD
END-IF.
*
* GetFirst
*
*
MOVE 40   TO BUF-LEN.
MOVE 0    TO KEY-NUMBER
CALL "_BTRV" USING B-GETFIRST, B-STATUS, POSITION-BLOCK,
                DATA-BUFFER, BUF-LEN, FILE-NAME, KEY-NUMBER.
IF B-STATUS NOT = 0
    DISPLAY 'Error Getting first record.Status= ' B-STATUS
ELSE
    DISPLAY 'Retrieved:' DECIMAL-FIELD STRING-FIELD
END-IF.
```

```
*
* TEST.BTR への更新
*
*
      MOVE 2      TO DECIMAL-FIELD.
      MOVE 'Record 2' TO STRING-FIELD.
      MOVE 40     TO BUF-LEN.
      MOVE 0      TO KEY-NUMBER
      CALL "_BTRV" USING B-UPDATE, B-STATUS, POSITION-BLOCK,
                    DATA-BUFFER, BUF-LEN, FILE-NAME, KEY-NUMBER.
      IF B-STATUS NOT = 0
          DISPLAY 'Error opening file.Status= ' B-STATUS
      ELSE
          DISPLAY 'Updated to:' DECIMAL-FIELD STRING-FIELD
      END-IF.
*
* TEST.BTR を閉じる
*
*
      MOVE 0      TO BUF-LEN.
      MOVE 0      TO KEY-NUMBER
      CALL "_BTRV" USING B-CLOSE, B-STATUS, POSITION-BLOCK,
                    DATA-BUFFER, BUF-LEN, FILE-NAME, KEY-NUMBER.
      IF B-STATUS NOT = 0
          DISPLAY 'Error closing file.Status= ' B-STATUS
      ELSE
          DISPLAY 'Successfully closed TEST.BTR'
      END-IF.

      STOP RUN.
```

Delphi

ここでは、以下の項目について説明します。

- 「プログラムの例」
- 「サンプルプログラムのコンパイル、リンク、実行」

プログラムの例

Web ダウンロードにより提供される以下のプログラム例では、Btrieve の一般的な操作方法を説明します。これらの操作は、MicroKernel との依存関係で要求される順番（ファイルを開いてから I/O を実行するなど）で行われます。

Btrsam32.pas

```
{*****
**
** Copyright 1982-2003 Pervasive Software Inc. All Rights Reserved
**
*****}
{*****}
```

BTRSAM32.DPR

これは、目的の環境で、コンパイラ ツールを使用した Btrieve アプリケーションのコンパイル、リンク、実行が可能なことを確認するための簡単なサンプルです。

このプログラムでは、Btrieve の Delphi インターフェイスを示します。
MS Windows NT/2000 および Windows 9x/ME 上の Delphi 2.0、3.0、4.0 および 5.0 で動作します。

このプログラムでは、サンプル ファイルで以下の操作を行います。

- BTRVID を使用して Microkernel Database エンジンのバージョン情報を取得する
- sample.btr を開く
- Key 0 の既知の値を持つレコードを取得する
- 読み込んだレコードを表示する
- Stat オペレーションを実行する
- 空の sample.btr のクローンを作成して開く
- Get Next Extended オペレーションを実行して sample.btr のレコードのサブセットを抽出する
- クローン ファイルにこれらのレコードを挿入する
- 両ファイルを閉じる

重要：Btrieve データ ファイルのサンプル sample.btr が保存されているディレクトリは、フルパスを指定してください。また、それぞれの「重要」項目も参考してください。

Delphi 2.0/3.0/4.0/5.0 Btrieve プロジェクトは、
Delphi プロジェクト環境のプルダウン メニューから以下を選択後コンパイルします。

```

PROJECT
  OPTIONS...
    COMPILER
      CODE GENERATION
        ALIGNED RECORD FIELDS ( de-select this )

```

これを行わない場合、レコードが正しく出力されません。
これは、レコードがバイトパックされていないためです。

代わりに、(*A-*) コンパイラ命令を使用するか、下に示すように、
すべてのレコードを「パック」として宣言します。
詳細については、Delphi のマニュアルを参照してください。

```

PROJECT FILES:
- btr32.dpr          Borland project file
- btr32.dof          Borland project file
- btrsam32.dfm       Borland project file
- btrsam32.pas       Source code for the simple sample
- btrapi32.pas       Delphi interface to Btrieve
- btrconst.pas       Btrieve constants file

```

```

*****}
unit btrsam32;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs,
  StdCtrls, BtrConst, BtrAPI32;

{*****}
  プログラム定数
{*****}
const
  { プログラム定数 }
  MY_THREAD_ID      = 50;
  EXIT_WITH_ERROR   = 1;
  VERSION_OFFSET    = 0;
  REVISION_OFFSET   = 2;
  PLATFORM_ID_OFFSET = 4;

{*****}
  重要：以下を変更し、使用する sample.btr へのフルパスを指定してください。
{*****}
  FILE_1            = 'c:\samples%sample.btr';
  FILE_2            = 'c:\samples%sample2.btr';

{*****}

```

Btrieve API の言語インターフェイス

```
バージョン処理用のレコードの型定義
*****}
type
  CLIENT_ID = packed record
    networkandnode :array[1..12] of char;
    applicationID  :array[1..3]  of char;
    threadID       :smallint;
  end;

  VERSION_STRUCT = packed record
    version  :smallint;
    revision :smallint;
    MKDEID   :char;
  end;

{*****
  sample.btr のレコードの定義
  *****}

{* writeln() への対応のため、ゼロ基準のキャラクター配列を使用 *}
PERSON_STRUCT = packed record
  ID          :longint;
  FirstName   :array[0..15] of char;
  LastName    :array[0..25] of char;
  Street      :array[0..30] of char;
  City        :array[0..30] of char;
  State       :array[0..2]  of char;
  Zip         :array[0..10] of char;
  Country     :array[0..20] of char;
  Phone       :array[0..13] of char;
end;

{*****
  Stat および Create 処理用のレコードの型定義
  *****}

FILE_SPECS = packed record
  recLength  :smallint;
  pageSize  :smallint;
  indexCount :smallint;
  reserved   :array[0..3] of char;
  flags      :smallint;
  dupPointers :byte;
  notUsed    :byte;
  allocations :smallint;
end;

KEY_SPECS = packed record
  position  :smallint;
  length    :smallint;
  flags     :smallint;
```

```

    reserved      :array[0..3] of char;
    keyType       :char;
    nullChar      :char;
    notUsed       :array[0..1] of char;
    manualKeyNumber :byte;
    acsNumber     :byte;
end;

FILE_CREATE_BUF = packed record
    fileSpecs :FILE_SPECS;
    keySpecs  :array[0..4] of KEY_SPECS;
end;

{*****
  Get Next Extended 処理用のレコードの型定義
*****}

GNE_HEADER = packed record
    descriptionLen :smallint;
    currencyConst  :array[0..1] of char;
    rejectCount    :smallint;
    numberTerms    :smallint;
end;

TERM_HEADER = packed record
    fieldType      :byte;
    fieldLen       :smallint;
    fieldOffset    :smallint;
    comparisonCode :byte;
    connector      :byte;
    value          :array[0..2] of char;
end;

RETRIEVAL_HEADER = packed record
    maxRecsToRetrieve :smallint;
    noFieldsToRetrieve :smallint;
end;

FIELD_RETRIEVAL_HEADER = packed record
    fieldLen       :smallint;
    fieldOffset    :smallint;
end;

PRE_GNE_BUFFER = packed record
    gneHeader :GNE_HEADER;
    term1     :TERM_HEADER;
    term2     :TERM_HEADER;
    retrieval :RETRIEVAL_HEADER;
    recordRet :FIELD_RETRIEVAL_HEADER;
end;

```

Btrieve API の言語インターフェイス

```
RETURNED_REC = packed record
    recLen      :smallint;
    recPos      :longint;
    personRecord :PERSON_STRUCT;
end;

POST_GNE_BUFFER = packed record
    numReturned :smallint;
    recs         :packed array[0..19] of RETURNED_REC;
end;

GNE_BUFFER_PTR = ^GNE_BUFFER;
GNE_BUFFER = packed record
case byte of
    1 : (preBuf  :PRE_GNE_BUFFER);
    2 : (postBuf :POST_GNE_BUFFER);
end;

{ *****
  Delphi で作成されたフォームの定義
  ***** }
TForm1 = class(TForm)
    RunButton  :TButton;
    ExitButton :TButton;
    ListBox1   :TListBox;
    procedure FormCreate(Sender:TObject);
    procedure ExitButtonClick(Sender:TObject);
    procedure RunButtonClick(Sender:TObject);
private
    { Private 宣言 }
    ArrowCursor,
    WaitCursor   :HCursor;
    status       :smallint;
    bufferLength :smallint;
    personRecord :PERSON_STRUCT;
    recordsRead  :longint;
    procedure RunTest;
public
    { Public 宣言 }
end;

var
    Form1:TForm1;

{ *****
  ここからプログラム開始
  ***** }
implementation
```

```

{$R *.DFM}

{*****
 プログラム変数
*****}
var
  { Btrieve 関数パラメーター }
  posBlock1      :string[128];
  posBlock2      :string[128];
  dataBuffer     :array[0..255] of char;
  dataLen        :word;
  keyBuf1        :string[255];
  keyBuf2        :string[255];
  keyNum         :smallint;

  btrieveLoaded  :boolean;
  personID       :longint;
  file1Open      :boolean;
  file2Open      :boolean;
  status         :smallint;
  getStatus      :smallint;
  i              :smallint;
  posCtr         :smallint;
  client         :CLIENT_ID;
  versionBuffer  :array[1..3] of VERSION_STRUCT;
  fileCreateBuf  :FILE_CREATE_BUF;
  gneBuffer      :GNE_BUFFER_PTR;
  personRecord   :PERSON_STRUCT;

{*****
 Listbox に書き込むためのヘルパー関数
*****}
procedure WritelnLB( LB:TListBox; Str:String);
begin
  LB.Items.Add(Str);
end;

procedure TForm1.FormCreate(Sender:TObject);
begin
  ArrowCursor    := LoadCursor(0, IDC_ARROW);
  WaitCursor     := LoadCursor(0, IDC_WAIT);
end;

{*****
 サンプルのメイン プロシージャ
*****}
procedure TForm1.RunTest;
begin
  ListBox1.Clear;

```

Btrieve API の言語インターフェイス

```
WritelnLB( ListBox1, 'Test started ...' );

{ 変数の初期化 }
btrieveLoaded := FALSE;
file1Open := FALSE;
file2Open := FALSE;
keyNum := 0;
status := B_NO_ERROR;
getStatus := B_NO_ERROR;

{ クライアント ID のセットアップ }
fillchar(client.networkAndNode, sizeof(client.networkAndNode), #0);
client.applicationID := 'MT' * #0; { "AA" 以上を設定 }
client.threadID := MY_THREAD_ID;

fillchar(versionBuffer, sizeof(versionBuffer), #0);
dataLen := sizeof(versionBuffer);

status = BTRVID(
    B_VERSION,
    posBlock1,
    versionBuffer,
    dataLen,
    keyBuf1[1],
    keyNum,
    client);

if status = B_NO_ERROR then begin
    writelnLB( ListBox1, '返された Btrieve バージョンは ' );
    for i := 1 to 3 do begin
        with versionBuffer[i] do begin
            if (version > 0) then begin
                writelnLB(ListBox1, intToStr(version) + '.' +
                    intToStr(revision) + ' ' + MKDEID);
            end
        end
    end;
    btrieveLoaded := TRUE;
end else begin
    writelnLB(ListBox1, 'Btrieve B_VERSION status = ' +
        intToStr(status));
    if status <> B_RECORD_MANAGER_INACTIVE then begin
        btrieveLoaded := TRUE;
    end
end;

{ sample.btr を開く }
if status = B_NO_ERROR then begin
    fillchar(dataBuffer, sizeof(dataBuffer), #0);
    fillchar(keyBuf1, sizeof(keyBuf1), #0);
```

```

keyNum := 0;
dataLen := 0;

keyBuf1 := FILE_1 + #0;
keyBuf2 := FILE_2 + #0;

status = BTRVID(
    B_OPEN,
    posBlock1,
    dataBuffer,
    dataLen,
    keyBuf1[1],
    keyNum,
    client);

writelnLB(ListBox1, 'Btrieve B_OPEN status = ' + intToStr(status));
if status = B_NO_ERROR then begin
    file1Open := TRUE;
end
end;

(* B_GET_EQUAL を使用して key 0 が既知の値であるレコードを取得 *)
if status = B_NO_ERROR then begin
    fillchar(personRecord, sizeof(personRecord), #0);
    dataLen := sizeof(personRecord);
    personID := 263512477;  (* 実際これは社会保障番号 *)

    status = BTRVID(
        B_GET_EQUAL,
        posBlock1,
        personRecord,
        dataLen,
        personID,
        keyNum,
        client);

    writelnLB(ListBox1, 'Btrieve B_GET_EQUAL status = ' +
intToStr(status));
    if status = B_NO_ERROR then with personRecord do begin
        writelnLB(ListBox1, '');
        writelnLB(ListBox1, 'Selected fields from the retrieved record
are:');
        writelnLB(ListBox1, 'ID:' + intToStr(ID));
        writelnLB(ListBox1, 'Name:' + FirstName + ' ' + LastName);
        writelnLB(ListBox1, 'Street:' + Street);
        writelnLB(ListBox1, 'City:' + City);
        writelnLB(ListBox1, 'State:' + State);
        writelnLB(ListBox1, 'Zip:' + Zip);
        writelnLB(ListBox1, 'Country:' + Country);
        writelnLB(ListBox1, 'Phone:' + Phone);
    end;
end;

```

Btrieve API の言語インターフェイス

```
writelnLB(ListBox1, '');
end;
end;

{ Stat オペレーションを実行し、CreateBuf (作成バッファ) を取得 }
fillchar(fileCreateBuf, sizeof(fileCreateBuf), #0);
dataLen := sizeof(fileCreateBuf);
keyNum := -1;
status := BTRVID(B_STAT,
                 posBlock1,
                 fileCreateBuf,
                 dataLen,
                 keyBuf1[1],
                 keyNum,
                 client);

if (status = B_NO_ERROR) then begin
  { sample2.btr を作成して開く }
  keyNum := 0;
  dataLen := sizeof(fileCreateBuf);
  status := BTRVID(B_CREATE,
                  posBlock2,
                  fileCreateBuf,
                  dataLen,
                  keyBuf2[1],
                  keyNum,
                  client);

  writelnLB(ListBox1, 'Btrieve B_CREATE status = ' +
intToStr(status));
end;

if (status = B_NO_ERROR) then begin
  keyNum := 0;
  dataLen := 0;

  status = BTRVID(
    B_OPEN,
    posBlock2,
    dataBuffer,
    dataLen,
    keyBuf2[1],
    keyNum,
    client);
  writelnLB(ListBox1, 'Btrieve B_OPEN status = ' + intToStr(status));
  if (status = B_NO_ERROR) then begin
    file2Open := TRUE;
  end;
end;
end;
```

```

{ 元のファイルからデータを抽出し、新しいファイルへ挿入 }
if (status = B_NO_ERROR) then begin
  { getFirst を実行してカレンシーを確立 }
  keyNum := 2; { STATE-CITY index }
  fillchar(personRecord, sizeof(personRecord), #0);
  fillchar(keyBuf1, sizeof(keyBuf1), #0);
  dataLen := sizeof(personRecord);

  getStatus := BTRVID(
    B_GET_FIRST,
    posBlock1,
    personRecord,
    dataLen,
    keyBuf1[1],
    keyNum,
    client);

  writelnLB(ListBox1, 'Btrieve B_GET_FIRST status = ' +
intToStr(GETstatus));
  writelnLB(ListBox1, '');
end;

{ ヒープ メモリの割り当て }
gneBuffer := new(GNE_BUFFER_PTR);
fillchar(gneBuffer^, sizeof(GNE_BUFFER), #0);

strPCopy(gneBuffer^.preBuf.gneHeader.currencyConst, 'UC');
while (getStatus = B_NO_ERROR) do begin
  gneBuffer^.preBuf.gneHeader.rejectCount := 0;
  gneBuffer^.preBuf.gneHeader.numberTerms := 2;
  posCtr := sizeof(GNE_HEADER);

  { 最初の条件を代入 }
  gneBuffer^.preBuf.term1.fieldType := 11;
  gneBuffer^.preBuf.term1.fieldLen := 3;
  gneBuffer^.preBuf.term1.fieldOffset := 108;
  gneBuffer^.preBuf.term1.comparisonCode := 1;
  gneBuffer^.preBuf.term1.connector := 2;

  strPCopy(gneBuffer^.preBuf.term1.value, 'TX');
  inc(posCtr, (sizeof(TERM_HEADER)));

  { 2 つ目の条件を代入 }
  gneBuffer^.preBuf.term2.fieldType := 11;
  gneBuffer^.preBuf.term2.fieldLen := 3;
  gneBuffer^.preBuf.term2.fieldOffset := 108;
  gneBuffer^.preBuf.term2.comparisonCode := 1;
  gneBuffer^.preBuf.term2.connector := 0;
  strPCopy(gneBuffer^.preBuf.term2.value, 'CA');
  inc(posCtr, sizeof(TERM_HEADER));

```

Btrieve API の言語インターフェイス

```
{ プロジェクション ヘッダーを設定してレコード全体を読み込む }
gneBuffer^.preBuf.retrieval.maxRecsToRetrieve := 20;
gneBuffer^.preBuf.retrieval.noFieldsToRetrieve := 1;
inc(posCtr, sizeof(RETRIEVAL_HEADER));
gneBuffer^.preBuf.recordRet.fieldLen := sizeof(PERSON_STRUCT);
gneBuffer^.preBuf.recordRet.fieldOffset := 0;
inc(posCtr, sizeof(FIELD_RETRIEVAL_HEADER));
gneBuffer^.preBuf.gneHeader.descriptionLen := posCtr;

dataLen := sizeof(GNE_BUFFER);
getStatus := BTRVID(
    B_GET_NEXT_EXTENDED,
    posBlock1,
    gneBuffer^,
    dataLen,
    keyBuf1,
    keyNum,
    client);

writelnLB(ListBox1, 'Btrieve B_GET_NEXT_EXTENDED status = ' +
intToStr(getStatus));

{ Get Next Extended は、ファイルの終わりまで到達してもレコードを }
{ 返すことがあります }
if ((getStatus = B_NO_ERROR) or (getStatus = B_END_OF_FILE)) then
begin
    writelnLB(ListBox1, 'GetNextExtended returned ' +
        intToStr(gneBuffer^.postBuf.numReturned) + ' records. ');
    for i := 0 to gneBuffer^.postBuf.numReturned - 1 do begin
        dataLen := sizeof(PERSON_STRUCT);
        personRecord := gneBuffer^.postBuf.recs[i].personRecord;
        status = BTRVID(
            B_INSERT,
            posBlock2,
            personRecord,
            dataLen,
            keyBuf2,
            -1, { カレンシー変更なし }
            client);
        if (status <> B_NO_ERROR) then begin
            writelnLB(ListBox1, 'Btrieve B_INSERT status = ' +
intToStr(status));
            break;
        end;
    end;
end;

writelnLB(ListBox1, 'Inserted ' +
intToStr(gneBuffer^.postBuf.numReturned) +
    ' records in new file, status = ' + intToStr(status));
```

```

        writelnLB(ListBox1, '');
    end;
    fillchar(gneBuffer^, sizeof(GNE_BUFFER), #0);
    gneBuffer^.preBuf.gneHeader.currencyConst := 'EG';
end;
dispose(gneBuffer);

{ 開いているファイルを閉じる }
keyNum := 0;
if file1Open = TRUE then begin
    dataLen := 0;

    status = BTRVID(
        B_CLOSE,
        posBlock1,
        dataBuffer,
        dataLen,
        keyBuf1[1],
        keyNum,
        client);

    writelnLB(ListBox1, 'Btrieve B_CLOSE status (sample.btr) = ' +
        intToStr(status));
    end;

if file2Open = TRUE then begin
    dataLen := 0;

    status = BTRVID(
        B_CLOSE,
        posBlock2,
        dataBuffer,
        dataLen,
        keyBuf2[1],
        keyNum,
        client);

    writelnLB(ListBox1, 'Btrieve B_CLOSE status (sample2.btr) = ' +
        intToStr(status));
    end;

{ リソース解放 }
dataLen := 0;
status := BTRVID(B_STOP, posBlock1, DataBuffer,
    dataLen, keyBuf1[1], 0, client);
writelnLB(ListBox1, 'Btrieve B_STOP status = ' + intToStr(status) );

end;

procedure TForm1.ExitButtonClick(Sender:TObject);

```

Btrieve API の言語インターフェイス

```
begin
  Close;
end;

procedure TForm1.RunButtonClick(Sender:TObject);
begin
  SetCursor(WaitCursor);
  RunTest;
  SetCursor(ArrowCursor);
end;

end.
```

サンプルプログラムのコンパイル、リンク、実行

➤Delphi 3、4、5 では、サンプルプログラムのコンパイル、リンク、実行を、以下の方法で行います。

- 1 [ファイル] メニューの [プロジェクトを開く] を選択し、¥Int¥Delphi ディレクトリの Btr32.dpr プロジェクト ファイルを開きます。
- 2 BtrSam32.pas ファイルで、Sample.btr および Sample2.btr のパスを適切なものに変更します。
- 3 ツールバーの [実行] ボタンをクリックします。
サンプルプログラムがコンパイル、リンク、実行されます。
- 4 [Btrieve Sample Application] ウィンドウの [Run Test] ボタンをクリックします。
サンプルプログラムが、Btrieve エンジンに対してテストを行います。

➤Delphi 1 では、サンプルプログラムのコンパイル、リンク、実行を、以下の方法で行います。

- 1 [File] メニューの [Open Project] を選択し、¥Int¥Delphi ディレクトリに Btr16.dpr プロジェクト ファイルを開きます。
- 2 BtrSam16.pas ファイルで、Sample.btr および Sample2.btr のパスを適切なものに変更します。
- 3 ツールバーの [実行] ボタンをクリックします。
サンプルプログラムがコンパイル、リンク、実行されます。
- 4 [Btrieve Sample Application] ウィンドウの [Run Test] ボタンをクリックします。
プログラムが、Btrieve エンジンに対してテストを行います。

Pascal

ここでは、以下の項目について説明します。

- 「プログラムの例」
- 「Pascal インターフェイスを使用したコンパイルとリンク」

プログラムの例

Web ダウンロードにより提供される以下のプログラム例では、Btrieve の一般的な操作方法を説明します。これらの操作は、MicroKernel との依存関係で要求される順番（ファイルを開いてから I/O を実行するなど）で行われます。

Btrsampw.pas

```
{*****
**
** Copyright 2003 Pervasive Software Inc. All Rights Reserved
**
*****}
{*****}
BTRSAMPW.PAS
このプログラムでは、Borland Pascal バージョン 7.0 または Turbo Pascal
for Windows バージョン 1.5 を使用して、MS Windows 対応の Btrieve
インターフェイスについて説明します。

このプログラムでは、サンプル ファイルで以下の操作を行います。
- BTRVID を使用して Microkernel Database エンジンのバージョン情報を
  取得する
- sample.btr を開く
- Key 0 の既知の値を持つレコードを取得する
- 読み込んだレコードを表示する
- Stat オペレーションを実行する
- 空の sample.btr のクローンを作成して開く
- Get Next Extended を実行して sample.btr のレコードのサブセットを
  抽出する
- クローン ファイルにこれらのレコードを挿入する
- 両ファイルを閉じる

重要：Btrieve データ ファイルのサンプル sample.btr が保存されている
ディレクトリへのフルパスを指定してください。また、それぞれの「重要」項目も
参考にしてください。
*****}
program btrsampw;

uses
  WinCrt,      { text mode I/O library for Windows }
```

Btrieve API の言語インターフェイス

```
Strings,      { Pascal System functions }
btrapiw,      { btrieve interface unit }
btrconst;    { Btrieve Constants Unit }

const
  {*****
   重要：以下を変更し、使用する sample.btr へのフルパスを指定してください。
   *****/}
  FILE1_NAME      = 'c:\samples\sample.btr' + #0;
  FILE2_NAME      = 'c:\samples\sample2.btr' + #0;

  { プログラム定数 }
  MY_THREAD_ID    = 50;
  EXIT_WITH_ERROR = 1;
  VERSION_OFFSET  = 0;
  REVISION_OFFSET = 2;
  PLATFORM_ID_OFFSET = 4;

  {*****
   バージョン処理用のレコードの型定義
   *****/}
type
  CLIENT_ID = packed record
    networkandnode :array[0..11] of char;
    applicationID  :array[0..1]  of char;
    threadID       :integer;
  end;

  VERSION_STRUCT = packed record
    version  :integer;
    revision :integer;
    MKDEId   :char;
  end;

  {*****
   sample.btr からのレコードの定義
   *****/}

  {* writeln() への対応のため、ゼロ基準のキャラクター配列を使用 *}
  PERSON_STRUCT = packed record
    ID          :longint;
    FirstName   :array[0..15] of char;
    LastName    :array[0..25] of char;
    Street      :array[0..30] of char;
    City        :array[0..30] of char;
    State       :array[0..2]  of char;
    Zip         :array[0..10] of char;
    Country     :array[0..20] of char;
    Phone       :array[0..13] of char;
  end;
```

```

{*****
  Stat および Create 処理用のレコードの型定義
*****}
FILE_SPECS = packed record
  recLength      :integer;
  pageSize      :integer;
  indexCount     :integer;
  reserved       :array[0..3] of char;
  flags         :integer;
  dupPointers   :byte;
  notUsed       :byte;
  allocations    :integer;
end;

KEY_SPECS = packed record
  position      :integer;
  length        :integer;
  flags         :integer;
  reserved      :array[0..3] of char;
  keyType       :char;
  nullChar      :char;
  notUsed       :array[0..1] of char;
  manualKeyNumber :byte;
  acsNumber     :byte;
end;

FILE_CREATE_BUF = packed record
  fileSpecs    :FILE_SPECS;
  keySpecs     :array[0..4] of KEY_SPECS;
end;

{*****
  Get Next Extended 処理用のレコードの型定義
*****}

GNE_HEADER = packed record
  descriptionLen :integer;
  currencyConst  :array[0..1] of char;
  rejectCount    :integer;
  numberTerms    :integer;
end;

TERM_HEADER = packed record
  fieldType     :byte;
  fieldLen      :integer;
  fieldOffset    :integer;
  comparisonCode :byte;
  connector     :byte;
  value         :array[0..2] of char;

```

Btrieve API の言語インターフェイス

```
end;

RETRIEVAL_HEADER = packed record
    maxRecsToRetrieve    :integer;
    noFieldsToRetrieve   :integer;
end;

FIELD_RETRIEVAL_HEADER = packed record
    fieldLen             :integer;
    fieldOffset          :integer;
end;

PRE_GNE_BUFFER = packed record
    gneHeader :GNE_HEADER;
    term1     :TERM_HEADER;
    term2     :TERM_HEADER;
    retrieval :RETRIEVAL_HEADER;
    recordRet :FIELD_RETRIEVAL_HEADER;
end;

RETURNED_REC = packed record
    recLen      :integer;
    recPos      :longint;
    personRecord :PERSON_STRUCT;
end;

POST_GNE_BUFFER = packed record
    numReturned :integer;
    recs         :packed array[0..19] of RETURNED_REC;
end;

GNE_BUFFER_PTR = ^GNE_BUFFER;
GNE_BUFFER = packed record
case byte of
    1 : (preBuf  :PRE_GNE_BUFFER);
    2 : (postBuf :POST_GNE_BUFFER);
end;

{*****
 変数
*****}
var
    { Btrieve 関数パラメーター }
    posBlock1 :string[128];
    posBlock2 :string[128];
    dataBuffer :array[0..255] of char;
    dataLen    :word;
    keyBuf1    :string[255];
    keyBuf2    :string[255];
    keyNum     :integer;
```

```

btrieveLoaded :boolean;
personID      :longint;
file1Open    :boolean;
file2Open    :boolean;
status       :integer;
getStatus    :integer;
i            :integer;
posCtr       :integer;

client       :CLIENT_ID;
versionBuffer :array[1..3] of VERSION_STRUCT;
fileCreateBuf :FILE_CREATE_BUF;
gneBuffer    :GNE_BUFFER_PTR;
personRecord :PERSON_STRUCT;

{*****
  ここからプログラム開始
  *****/}
begin { btrsamp }
  { 変数の初期化 }
  btrieveLoaded := FALSE;
  file1Open := FALSE;
  file2Open := FALSE;
  keyNum := 0;
  status := B_NO_ERROR;
  getStatus := B_NO_ERROR;

  writeln;
  writeln('***** Btrieve Pascal Interface for Windows Demo
  *****');
  writeln;

  { クライアント ID のセットアップ }
  fillchar(client.networkAndNode, sizeof(client.networkAndNode), #0);

{$ifdef ver70} {Note:Delphi 1.0 is ver80}
  client.applicationID := 'MT' + #0; { "AA" 以上を設定 }
{$else}
  strcpy(client.applicationID, 'MT'); { "AA" 以上を設定 }
  strcat(client.applicationID, #0);
{$endif}

  client.threadID := MY_THREAD_ID;

  fillchar(versionBuffer, sizeof(versionBuffer), #0);
  dataLen := sizeof(versionBuffer);

  status = BTRVID(
    B_VERSION,

```

Btrieve API の言語インターフェイス

```
        posBlock1,
        versionBuffer,
        dataLen,
        keyBuf1[1],
        keyNum,
        client);

if status = B_NO_ERROR then begin
    writeln('Btrieve Versions returned are:');
    for i := 1 to 3 do begin
        with versionBuffer[i] do begin
            if (version > 0) then begin
                writeln(version, '.', revision, ' ', MKDEId);
            end
        end
    end
end;
btrieveLoaded := TRUE;
end else begin
    writeln('Btrieve B_VERSION status = ', status);
    if status <> B_RECORD_MANAGER_INACTIVE then begin
        btrieveLoaded := TRUE;
    end
end;

(* sample.btr を開く *)
if status = B_NO_ERROR then begin
    fillchar(dataBuffer, sizeof(dataBuffer), #0);
    fillchar(keyBuf1, sizeof(keyBuf1), #0);
    keyNum := 0;
    dataLen := 0;

    keyBuf1 := FILE1_NAME;
    keyBuf2 := FILE2_NAME;

    status = BTRVID(
        B_OPEN,
        posBlock1,
        dataBuffer,
        dataLen,
        keyBuf1[1],
        keyNum,
        client);

    writeln('Btrieve B_OPEN status = ', status);
    if status = B_NO_ERROR then begin
        file1Open := TRUE;
    end
end;
end;
```

```

{* B_GET_EQUAL を使用して key 0 が既知の値であるレコードを取得 *}
if status = B_NO_ERROR then begin
    fillchar(personRecord, sizeof(personRecord), #0);
    dataLen := sizeof(personRecord);
    personID := 263512477;  {* 実際これは社会保障番号 *}

    status = BTRVID(
        B_GET_EQUAL,
        posBlock1,
        personRecord,
        dataLen,
        personID,
        keyNum,
        client);

    writeln('Btrieve B_GET_EQUAL status = ', status);
    if status = B_NO_ERROR then with personRecord do begin
        writeln;
        writeln('Selected fields from the retrieved record are:');
        writeln('ID:', ID);
        writeln('Name:', FirstName, ' ', LastName);
        writeln('Street:', Street);
        writeln('City:', City);
        writeln('State:', State);
        writeln('Zip:', Zip);
        writeln('Country:', Country);
        writeln('Phone:', Phone);
        writeln;
    end;
end;

{ Stat オペレーションを実行し、CreateBuf (作成バッファ) を取得 }
fillchar(fileCreateBuf, sizeof(fileCreateBuf), #0);
dataLen := sizeof(fileCreateBuf);
keyNum := -1;
status := BTRVID(B_STAT,
    posBlock1,
    fileCreateBuf,
    dataLen,
    keyBuf1[1],
    keyNum,
    client);

if (status = B_NO_ERROR) then begin
    { sample2.btr を作成して開く }
    keyNum := 0;
    dataLen := sizeof(fileCreateBuf);
    status := BTRVID(B_CREATE,
        posBlock2,
        fileCreateBuf,

```

Btrieve API の言語インターフェイス

```
        dataLen,  
        keyBuf2[1],  
        keyNum,  
        client);  
  
    writeln('Btrieve B_CREATE status = ', status);  
end;  
  
if (status = B_NO_ERROR) then begin  
    keyNum := 0;  
    dataLen := 0;  
  
    status = BTRVID(  
        B_OPEN,  
        posBlock2,  
        dataBuffer,  
        dataLen,  
        keyBuf2[1],  
        keyNum,  
        client);  
    writeln('Btrieve B_OPEN status (sample2.btr) = ', status);  
    if (status = B_NO_ERROR) then begin  
        file2Open := TRUE;  
    end;  
end;  
  
{ 元のファイルからデータを抽出し、新しいファイルへ挿入 }  
if (status = B_NO_ERROR) then begin  
    { getFirst を実行してカレンシーを確立 }  
    keyNum := 2; { STATE-CITY index }  
    fillchar(personRecord, sizeof(personRecord), #0);  
    fillchar(keyBuf1, sizeof(keyBuf1), #0);  
    dataLen := sizeof(personRecord);  
  
    getStatus := BTRVID(  
        B_GET_FIRST,  
        posBlock1,  
        personRecord,  
        dataLen,  
        keyBuf1[1],  
        keyNum,  
        client);  
  
    writeln('Btrieve B_GET_FIRST status (sample.btr) = ', getStatus);  
    writeln;  
end;  
  
if maxavail < SizeOf(GNE_BUFFER) then begin  
    writeln('Insufficient memory to allocate buffer');  
    halt(EXIT_WITH_ERROR);  
end;
```

```

end else begin
  { ヒープ メモリの割り当て }
  gneBuffer := new(GNE_BUFFER_PTR);
end;
fillchar(gneBuffer^, sizeof(GNE_BUFFER), #0);
strPCopy(gneBuffer^.preBuf.gneHeader.currencyConst, 'UC');
while (getStatus = B_NO_ERROR) do begin
  gneBuffer^.preBuf.gneHeader.rejectCount := 0;
  gneBuffer^.preBuf.gneHeader.numberTerms := 2;
  posCtr := sizeof(GNE_HEADER);

  { 最初の条件を代入 }
  gneBuffer^.preBuf.term1.fieldType := 11;
  gneBuffer^.preBuf.term1.fieldLen := 3;
  gneBuffer^.preBuf.term1.fieldOffset := 108;
  gneBuffer^.preBuf.term1.comparisonCode := 1;
  gneBuffer^.preBuf.term1.connector := 2;

  strPCopy(gneBuffer^.preBuf.term1.value, 'TX');
  inc(posCtr, (sizeof(TERM_HEADER)));

  { 2 つ目の条件を代入 }
  gneBuffer^.preBuf.term2.fieldType := 11;
  gneBuffer^.preBuf.term2.fieldLen := 3;
  gneBuffer^.preBuf.term2.fieldOffset := 108;
  gneBuffer^.preBuf.term2.comparisonCode := 1;
  gneBuffer^.preBuf.term2.connector := 0;
  strPCopy(gneBuffer^.preBuf.term2.value, 'CA');
  inc(posCtr, sizeof(TERM_HEADER));

  { プロジェクション ヘッダーを設定してレコード全体を読み込む }
  gneBuffer^.preBuf.retrieval.maxRecsToRetrieve := 20;
  gneBuffer^.preBuf.retrieval.noFieldsToRetrieve := 1;
  inc(posCtr, sizeof(RETRIEVAL_HEADER));
  gneBuffer^.preBuf.recordRet.fieldLen := sizeof(PERSON_STRUCT);
  gneBuffer^.preBuf.recordRet.fieldOffset := 0;
  inc(posCtr, sizeof(FIELD_RETRIEVAL_HEADER));
  gneBuffer^.preBuf.gneHeader.descriptionLen := posCtr;

  dataLen := sizeof(GNE_BUFFER);
  getStatus := BTRVID(
    B_GET_NEXT_EXTENDED,
    posBlock1,
    gneBuffer^,
    dataLen,
    keyBuf1,
    keyNum,
    client);

  writeln('Btrieve B_GET_NEXT_EXTENDED status = ', getStatus);

```

Btrieve API の言語インターフェイス

```
{ Get Next Extended は、ファイルの終わりまで到達してもレコードを }
{ 返すことがあります }
if ((getStatus = B_NO_ERROR) or (getStatus = B_END_OF_FILE)) then
begin
  writeln('GetNextExtended returned ',
gneBuffer^.postBuf.numReturned, ' records. ');
  for i := 0 to gneBuffer^.postBuf.numReturned - 1 do begin

{$ifdef ver70}
    dataLen := sizeof(PERSON_STRUCT);
    personRecord := gneBuffer^.postBuf.recs[i].personRecord;
    status = BTRVID(
      B_INSERT,
      posBlock2,
      personRecord,
      dataLen,
      keyBuf2,
      -1, { カレンシー変更なし }
      client);
    if (status <> B_NO_ERROR) then begin
      writeln('Btrieve B_INSERT status = ', status);
      break;
    end;

{$else} {Turbo Pascal for Windows 1.5 は break に対応していません }
    if (status = B_NO_ERROR) then begin
      dataLen := sizeof(PERSON_STRUCT);
      personRecord := gneBuffer^.postBuf.recs[i].personRecord;
      status = BTRVID(
        B_INSERT,
        posBlock2,
        personRecord,
        dataLen,
        keyBuf2,
        -1, { カレンシー変更なし }
        client);

      end;
      if (status <> B_NO_ERROR) then begin
        writeln('Btrieve B_INSERT status = ', status);
      end;
    end;

{$endif}

    end;

    writeln('Inserted ', gneBuffer^.postBuf.numReturned, ' records in
new file, status = ', status);
    writeln;
  end;
  fillchar(gneBuffer^, sizeof(GNE_BUFFER), #0);
```

```
    gneBuffer^.preBuf.gneHeader.currencyConst := 'EG';
end;
dispose(gneBuffer);

{ 開いているファイルを閉じる }
keyNum := 0;
if file1Open = TRUE then begin
    dataLen := 0;

    status = BTRVID(
        B_CLOSE,
        posBlock1,
        dataBuffer,
        dataLen,
        keyBuf1[1],
        keyNum,
        client);

    writeln('Btrieve B_CLOSE status (sample.btr) = ', status);
end;

if file2Open = TRUE then begin
    dataLen := 0;

    status = BTRVID(
        B_CLOSE,
        posBlock2,
        dataBuffer,
        dataLen,
        keyBuf2[1],
        keyNum,
        client);

    writeln('Btrieve B_CLOSE status (sample2.btr) = ', status);
end;

end.
```

Pascal インターフェイスを使用したコンパイルとリンク

このセクションでは、一般的な開発環境に共通する操作方法を説明します。お使いの開発環境が該当しない場合は、この操作方法を参考にしてください。

➤ Borland Pascal 7 for Windows を使用してサンプル プログラムをコンパイル、リンク、実行するには

- 1 [File] メニューの [Open] を選択し、BtrSampW.pas ファイルを開きます。
- 2 BtrSampW.pas ファイルで、Sample.btr および Sample2.btr のパスを適切なものに変更します。
- 3 [Compile] メニューの [Make] を選択します。[OK] をクリックします。サンプル プログラムがコンパイル、およびリンクされます。
- 4 [Run] メニューの [Run] を選択します。サンプル プログラムが実行されます。

➤ Borland Pascal 7 for DOS を使用してサンプル プログラムをコンパイル、リンク、実行するには

- 1 [File] メニューの [Open] を選択し、以下のファイルを開きます。
 - ¥Intf¥Pascal¥BtrConst.pas
 - ¥Intf¥Pascal¥BtrApiD.pas
 - ¥Intf¥Pascal¥BtrSampD.pas
- 2 BtrSampD.pas ファイルで、Sample.btr および Sample2.btr のパスを適切なものに変更します。
- 3 [Options] メニューの [Compiler] を選択し、[Conditional Defines] を "BTI_DOS" に設定します。
- 4 [Options] メニューの [Directories] を選択し、[EXE] および [TPU] を "¥Intf¥Pascal" ディレクトリに設定します。
- 5 BtrConst.pas ファイルで、[Compile] メニューの [Make] を選択します。
- 6 BtrApiD.pas ファイルで、[Compile] メニューの [Make] を選択します。
- 7 BtrSampD.pas ファイルで、[Compile] メニューの [Make] を選択します。
- 8 DOS プロンプトから BtrSampD.exe を実行します。



メモ サンプル プログラムでは、BtrvID 呼び出しが使用されるため、/T:1 フラグのある Btrieve DOS リクエスターをロードしてください。また、同時に適切なリクエスター（Windows サーバーでは BReqNT）もロードする必要があります。

➤ Borland Turbo Pascal 1.5 for Windows を使用してサンプル プログラムをコンパイル、リンク、実行するには

- 1 [File] メニューの [Open] を選択し、以下のファイルを開きます。
 - ¥Intf¥Pascal¥BtrConst.pas
 - ¥Intf¥Pascal¥BtrApiW.pas
 - ¥Intf¥Pascal¥BtrSampW.pas
- 2 BtrSampW.pas ファイルで、Sample.btr および Sample2.btr のパスを適切なものに変更します。
- 3 BtrConst.pas ファイルで、[Compile] メニューの [Make] を選択します。
- 4 BtrApiW.pas ファイルで、[Compile] メニューの [Make] を選択します。
- 5 BtrSampW.pas ファイルで、[Compile] メニューの [Make] を選択します。
- 6 [Run] メニューの [Run] を選択します。
サンプル プログラムが実行されます。

Visual Basic

ここでは、以下の項目について説明します。

- 「プログラムの例」
- 「サンプルプログラムのコンパイル、リンク、実行」

プログラムの例

以下のプログラムの例では、Btrieve の一般的な操作の実行方法をいくつか説明します。これらの操作は、MicroKernel との依存関係で要求される順番（ファイルを開いてから I/O を実行するなど）で行われます。この Pervasive PSQL アプリケーションのサンプルを実行するには、以下のファイルが必要になります。

- ◆ BTR32VB.bas
- ◆ BTR32VBFieldMap.bas
- ◆ btr32VbSample.vbp (プロジェクト ファイル)
- ◆ btrv32vb.frm
- ◆ btrv32vb.frx

BTR32VBFieldMap.bas

```
*****
' **
' ** Copyright 2003 Pervasive Software Inc. All Rights Reserved
' **
' *****
' *****
' **
' ** BTR32VBFieldMap.bas
' **
' ** このソフトウェアは、Pervasive ソフトウェア開発キット (SDK) の一部です。
' **
' ** このソース コードは、単なる Pervasive PSQL マニュアルの補足的なものです。
' ** Pervasive PSQL の使用に関する詳細については、
' ** Pervasive PSQL の該当するマニュアルを参照してください。
' **
' *****
Option Explicit
Declare Sub CopyMemory Lib "KERNEL32" Alias "RtlMoveMemory" (hpvDest As
Any, hpvSource As Any, ByVal cbCopy As Long)
' *****
' 32 ビット アプリケーションのコンパイルの際、Visual Basic により、ユーザー
' 定義型 (UDT) は、メンバーのサイズによって 8 ビット、16 ビット、32 ビットの
' バウンダリにアラインされます。構造体とは異なり、データベースの列はパックされて
' おり、フィールド間の未使用スペースがありません。このアライン機能をオフにする
' ことはできないため、Visual Basic アプリケーションがデータベースにアクセスする
```

```

' ための構造体をパック / アンパックする手段が必要になります。
' Pervasive Btrieve Alignment DLL または PALN32.DLL は、このアライメント
' 問題を解決するためのものです。
' *****
' *****
' Pervasive Btrieve Alignment ライブラリ (PALn32.DLL) は、Pervasive
' ソフトウェア開発キットに用意されています。このライブラリは、アライメント調整
' された構造体のパックや、データベースの行のアンパックに使用されます。
' *****
' まず、データベースにアクセスする前に、データを処理するためのデータ構造体が
' 必要になります。ここでは、データをホールドするための構造体を作成します。
' *****
' Person ポジション ブロック
' *****
Public Type PosBlock
    buf(0 To 127) As Byte
End Type
' *****
' バージョン処理用のレコードの型定義
' *****
Public Type Version_BaseStruct
    version    As Integer
    revision   As Integer
    MKDEID     As String * 1
End Type
Public Type Version_Struct
    ver(0 To 2) As Version_BaseStruct
End Type
Public Type Client_ID
    networkandnode(0 To 11) As Byte
    applicationID(0 To 2)   As Byte
    threadID                As Integer
End Type
' *****
' Stat および Create 処理用のレコードの種類定義
' *****
Public Type BtrFileSpec
    length      As Integer
    PageSize    As Integer
    NumIndexes  As Integer
    Reserved    As Long
    FileFlags   As Integer
    NumDupPtr   As Byte
    NotUsed     As Byte
    Allocation  As Integer
End Type
' *****
' sample.btr からのレコードの定義
' *****
Public Type PersonRecType

```

Btrieve API の言語インターフェイス

```

    ID           As Long
    FirstName    As String * 16
    LastName     As String * 26
    Street       As String * 31
    City         As String * 31
    State        As String * 3
    Zip          As String * 11
    Country      As String * 21
    Phone        As String * 14
End Type

'*****
'  Get Next Extended 処理用のレコードの型定義
'*****
Public Type GNE_HEADER
    descriptionLen As Integer
    currencyConst  As String * 2
    rejectCount    As Integer
    numberTerms    As Integer
End Type
Public Type TERM_HEADER
    fieldType      As Byte
    fieldLen       As Integer
    fieldOffset    As Integer
    comparisonCode As Byte
    connector       As Byte
    value          As String * 3
End Type

Public Type RETRIEVAL_HEADER
    maxRecsToRetrieve As Integer
    noFieldsToRetrieve As Integer
End Type
Public Type FIELD_RETRIEVAL_HEADER
    fieldLen      As Integer
    fieldOffset    As Integer
End Type

Public Type PRE_GNE_BUFFER
    gneHeader As GNE_HEADER
    term1      As TERM_HEADER
    term2      As TERM_HEADER
    retrieval  As RETRIEVAL_HEADER
    recordRet  As FIELD_RETRIEVAL_HEADER
End Type
Public Type RETURNED_REC
    recLen      As Integer
    recPos      As Long
    personRecord As PersonRecType
End Type
```

```

Public Type POST_GNE_BUFFER
    numReturned    As Integer
    recs(0 To 19)  As RETURNED_REC
End Type

' *****
' FieldMap の定義。保持しているファイル構造の構造体を作成します。
' *****
Global Version_StructMap(0 To 8) As FieldMap
Const Versionsize = 15
Global ClientIDFldMap(0 To 16) As FieldMap
Const ClientIDsize = 17
Global PersonFldMap(0 To 8) As FieldMap
Const PersonRowSize = 157
Global gneheaderMap(0 To 3) As FieldMap
Const gneheadersize = 8
Global termheaderMap(0 To 5) As FieldMap
Const termheadersize = 10
Global retrievalheaderMap(0 To 1) As FieldMap
Const retrievalheadersize = 4
Global fieldretrievalMap(0 To 1) As FieldMap
Const fieldretrievalsize = 4
Global pregnebufferMap(0 To 23) As FieldMap
Const pregnebuffersize = 36
Global returnrecMap(0 To 12) As FieldMap
Const returnrecsize = 163
Global Post_GNE_BUFFERFieldMap(0 To 281) As FieldMap
Const postgnebuffersize = 3262
' *****
' FldMapType の定義。パックされたデータベースの行を保存するために必要な
' 構造体を作成します。
' *****
Public Type PersonRowType
    buf(1 To PersonRowSize) As Byte
End Type
Public Type VersionType
    buf(1 To Versionsize) As Byte
End Type
Public Type ClientIDType
    buf(1 To ClientIDsize) As Byte
End Type
Public Type gneheaderType
    buf(1 To gneheadersize) As Byte
End Type
Public Type termheaderType
    buf(1 To termheadersize) As Byte
End Type
Public Type retrievalheadertype
    buf(1 To retrievalheadersize) As Byte
End Type

```

Btrieve API の言語インターフェイス

```
Public Type fieldretrievaltype
    buf(1 To fieldretrievalsize) As Byte
End Type
Public Type pregnebuffertype
    buf(1 To pregnebuffersize) As Byte
End Type
Public Type returnrectype
    buf(1 To returnrecsize) As Byte
End Type
Public Type postgnebuffertype
    buf(1 To postgnebuffersize) As Byte
End Type
Public Const FLD_PAD32 = 42
'*****
' FieldMap のビルド。ファイル構造をメモリに読み込みます。
'*****
Sub AddField(map() As FieldMap, ByRef ctr As Integer, dataType As Long, _
    length As Long)

    SetField map(ctr), dataType, length
    ctr = ctr + 1

End Sub
Sub AddgncHeaderFldMap(map() As FieldMap, ByRef ctr As Integer)

    AddField map, ctr, FLD_INTEGER, 2 ' descriptionLen
    AddField map, ctr, FLD_STRING, 2 ' currencyConst
    AddField map, ctr, FLD_INTEGER, 2 ' rejectCount
    AddField map, ctr, FLD_INTEGER, 2 ' numberTerms
End Sub
Sub AddTERM_HEADERFldMap(map() As FieldMap, ByRef ctr As Integer)

    AddField map, ctr, FLD_BYTE, 1 ' fieldType
    AddField map, ctr, FLD_INTEGER, 2 ' fieldLen
    AddField map, ctr, FLD_INTEGER, 2 ' fieldOffset
    AddField map, ctr, FLD_BYTE, 1 ' comparisonCode
    AddField map, ctr, FLD_BYTE, 1 ' connector
    AddField map, ctr, FLD_STRING, 3 ' value
End Sub
Sub AddPersonFieldMap(map() As FieldMap, ByRef ctr As Integer)

    AddField map, ctr, FLD_INTEGER, 4 ' ID
    AddField map, ctr, FLD_STRING, 16 ' FirstName
    AddField map, ctr, FLD_STRING, 26 ' LastName
    AddField map, ctr, FLD_STRING, 31 ' Street
    AddField map, ctr, FLD_STRING, 31 ' City
    AddField map, ctr, FLD_STRING, 3 ' State
    AddField map, ctr, FLD_STRING, 11 ' Zip
    AddField map, ctr, FLD_STRING, 21 ' Country
    AddField map, ctr, FLD_STRING, 14 ' Phone
```

```

End Sub
Sub AddFieldMap(out() As FieldMap, ByRef ctr As Integer, nin() As
FieldMap)
' fieldmap をほかに追加
Dim fld As Integer

    For fld = LBound(nin) To UBound(nin)
        out(ctr) = nin(fld)
        ctr = ctr + 1
    Next fld

End Sub
Sub AddRETRIEVAL_HEADER(map() As FieldMap, ByRef ctr As Integer)

    AddField map, ctr, FLD_INTEGER, 2 ' maxRecsToRetrieve
    AddField map, ctr, FLD_INTEGER, 2 ' noFieldsToRetrieve

End Sub
Sub AddFIELD_RETRIEVAL_HEADER(map() As FieldMap, ByRef ctr As Integer)

    AddField map, ctr, FLD_INTEGER, 2 ' fieldLen
    AddField map, ctr, FLD_INTEGER, 2 ' fieldOffset

End Sub
Sub AddPreGNEBufferFldMap(map() As FieldMap, ByRef ctr As Integer)

    AddgneHeaderFldMap map, ctr          ' gneHeader
    AddTERM_HEADERFldMap map, ctr        ' term1
    AddTERM_HEADERFldMap map, ctr        ' term2
    AddRETRIEVAL_HEADER map, ctr         ' retrieval
    AddFIELD_RETRIEVAL_HEADER map, ctr   ' recordRet

End Sub
Sub AddRETURNED_RECFLdMap(map() As FieldMap, ByRef ctr As Integer)
    AddField map, ctr, FLD_INTEGER, 2 ' recLen
    AddField map, ctr, FLD_INTEGER, 4 ' recPos
    AddFieldMap map, ctr, PersonFldMap ' personRecord

End Sub
Sub AddPostGNEBufferFldMap(map() As FieldMap, ByRef ctr As Integer)
Dim fld As Integer

    AddField map, ctr, FLD_INTEGER, 2 ' numReturned

    For fld = 0 To 19
        AddField map, ctr, FLD_PAD32, 0
        AddRETURNED_RECFLdMap map, ctr ' recs
    Next fld

```

Btrieve API の言語インターフェイス

```
End Sub
Sub AddVersionBufferFldMap(map() As FieldMap, ByRef ctr As Integer)
Dim fld As Integer
  For fld = 0 To 2
    AddField map, ctr, FLD_INTEGER, 2 ' version
    AddField map, ctr, FLD_INTEGER, 2 ' revision
    AddField map, ctr, FLD_STRING, 1 ' MKDEId
  Next fld
End Sub

Sub AddClientIDBufferFldMap(map() As FieldMap, ByRef ctr As Integer)
Dim fld As Integer
  For fld = 0 To 11
    AddField map, ctr, FLD_BYTE, 1 ' networkandnode
  Next fld

  For fld = 0 To 2
    AddField map, ctr, FLD_BYTE, 1 ' applicationID
  Next fld

  AddField map, ctr, FLD_INTEGER, 2 ' threadID
End Sub

Sub InitFieldMaps()
' FieldMaps の初期化
AddPersonFieldMap PersonFldMap, 0
AddgneHeaderFldMap gneheaderMap, 0
AddTERM_HEADERFldMap termheaderMap, 0
AddFIELD_RETRIEVAL_HEADER fieldretrievalMap, 0
AddRETRIEVAL_HEADER retrievalheaderMap, 0
AddRETURNED_RECFLdMap returnrecMap, 0
AddPreGNEBufferFldMap pregnebufferMap, 0
AddPostGNEBufferFldMap Post_GNE_BUFFERFieldMap, 0
AddVersionBufferFldMap Version_StructMap, 0
AddClientIDBufferFldMap ClientIDFldMap, 0

End Sub
BTR32VB.bas
' {*****}
' **
' ** Copyright 2003 Pervasive Software Inc. All Rights Reserved
' **
' {*****}
' {*****}
' **
' ** BTR32VB.bas
' **
' ** このソフトウェアは、Pervasive ソフトウェア開発キット (SDK) の一部です。
```

```

***
*** このソース コードは、単なる Pervasive PSQL マニュアルの補足的なものです。
*** Pervasive PSQL の使用に関する詳細については、
*** Pervasive PSQL の該当するマニュアルを参照してください。
***
*****}
' *****
'
'                               データ型
'
*****
Option Explicit
DefInt A-Z
Global Const BOPEN = 0
Global Const BCLOSE = 1
Global Const BINSERT = 2
Global Const BUPDATE = 3
Global Const BDELETE = 4
Global Const BGETEQUAL = 5
Global Const BGETNEXT = 6
Global Const BGETPREVIOUS = 7
Global Const BGETGREATEROREQUAL = 9
Global Const BGETFIRST = 12
Global Const BGETLAST = 13
Global Const BCREATE = 14
Global Const BSTAT = 15
Global Const BBEGINTRANS = 19
Global Const BTRANSEND = 20
Global Const BABORTTRANS = 21
Global Const BGETPOSITION = 22
Global Const BGETRECORD = 23
Global Const BSTOP = 25
Global Const BVERSION = 26
Global Const BRESET = 28
Global Const BGETNEXTEXTENDED = 36
Global Const BGETKEY = 50
Global Const KEY_BUF_LEN = 255
Rem キー フラグ
Global Const DUP = 1
Global Const MODIFIABLE = 2
Global Const BIN = 4
Global Const NUL = 8
Global Const SEGMENT = 16
Global Const SEQ = 32
Global Const DEC = 64
Global Const SUP = 128
Rem キー タイプ
Global Const EXTTYPE = 256
Global Const MANUAL = 512
Global Const BSTRING = 0
Global Const BINTEGER = 1

```

Btrieve API の言語インターフェイス

```
Global Const BFLOAT = 2
Global Const BDATE = 3
Global Const BTIME = 4
Global Const BDECIMAL = 5
Global Const BNUMERIC = 8
Global Const BZSTRING = 11
Global Const BAUTOINC = 15
Global Const B_NO_ERROR = 0
Global Const B_END_OF_FILE = 9
Global Const VAR_RECS = &H1
Global Const BLANK_TRUNC = &H2
Global Const PRE_ALLOC = &H4
Global Const DATA_COMP = &H8
Global Const KEY_ONLY = &H10
Global Const BALANCED_KEYS = &H20
Global Const FREE_10 = &H40
Global Const FREE_20 = &H80
Global Const FREE_30 = &HC0
Global Const DUP_PTRS = &H100
Global Const INCLUDE_SYSTEM_DATA = &H200
Global Const NO_INCLUDE_SYSTEM_DATA = &H1200
Global Const SPECIFY_KEY_NUMS = &H400
Global Const VATS_SUPPORT = &H800
Global Const FLD_STRING = 0
Global Const FLD_INTEGER = 1
Global Const FLD_IEEE = 2
Global Const FLD_DATE = 3
Global Const FLD_TIME = 4
Global Const FLD_MONEY = 6
Global Const FLD_LOGICAL = 7
Global Const FLD_BYTE = 19
Global Const FLD_UNICODE = 20
Declare Function BTRCALL Lib "w3btrv7.dll" (ByVal OP, Pb As Any, Db As
Any, DL As Long, ByRef Kb As Any, ByVal Kl, ByVal Kn) As Integer
Declare Function BTRCALLID Lib "w3btrv7.dll" (ByVal OP, Pb As Any, Db
As Any, DL As Long, Kb As Any, ByVal Kl, ByVal Kn, ID As Any) As Integer

Sub SetField(ByRef fld As FieldMap, dataType As Long, Size As Long)
    fld.dataType = dataType
    fld.Size = Size
End Sub
btrv32vb.frm
' {*****}
' **
' ** Copyright 2003 Pervasive Software Inc. All Rights Reserved
' **
' {*****}
' {*****}
' **
' **
```

```

'** btrv32vb.frm
'**
'** このソフトウェアは、Pervasive ソフトウェア開発キット (SDK) の一部です。
'**
'** このソース コードは、単なる Pervasive PSQL マニュアルの補足的なものです。
'** Pervasive PSQL の使用に関する詳細については、
'** Pervasive PSQL の該当するマニュアルを参照してください。
'**
'*****}
Option Explicit
Dim sPersonPosBlk As PosBlock      'Person ポジション ブロック
Dim sPersonPosBlk2 As PosBlock    'Person ポジション ブロック
Dim nPersonKeyNum As Integer      'Person インデックス番号
Dim nKeyBufLen As Integer         'キー バッファー長
Dim nKeyBufLen2 As Integer        'キー バッファー長
Dim sKeyBuffer As String          'Person テーブルのキー バッファー
Dim sKeyBuffer2 As String         'Person テーブルのキー バッファー
Dim NewFileSpec As BtrFileSpec   'ファイルの STAT 取得に使用
Dim PersonRow As PersonRowType    'BTR32VBFieldMap.bas で作成される種類
Private Sub cmdExit_Click()
    Unload Me
End Sub
'*****
' サンプルのメイン プロシージャ
'*****
Private Sub cmdRunTest_Click()
Dim lPersonID As Long
Dim recordaddress As Long
Dim prebuffer As PRE_GNE_BUFFER
Dim prebufftype As pregnebuffertype
Dim postbuffer As POST_GNE_BUFFER
Dim postbufftype As postgnebuffertype
Dim msg As String
Dim DataLen As Integer
Dim nStatus As Integer
Dim versionBuffer As VersionType
Dim versionstruct As Version_Struct
Dim i As Integer
Dim FileOpen As Boolean
Dim File2Open As Boolean
Dim personRecord As PersonRecType
Dim personRec As PersonRowType
Dim client As Client_ID
Dim clientrow As ClientIDType
Dim s As String
Dim s2 As String
Dim PosBlockSize As Integer
    PosBlockSize = 128

    sKeyBuffer = Space$(KEY_BUF_LEN)

```

Btrieve API の言語インターフェイス

```
sKeyBuffer2 = Space$(KEY_BUF_LEN)
nKeyBufLen = KEY_BUF_LEN
nKeyBufLen2 = KEY_BUF_LEN

s = String(PosBlockSize, 0)
s2 = String(PosBlockSize, 0)
CopyMemory sPersonPosBlk, s, PosBlockSize
CopyMemory sPersonPosBlk2, s2, PosBlockSize

' ユーザー定義パスを読み込む
sKeyBuffer = Trim(txtInput.Text)
sKeyBuffer2 = Trim(txtOutput.Text)

nPersonKeyNum = 0

'Version Btrieve 呼び出し

For i = 0 To 11
    client.networkandnode(i) = CByte(0)
Next i

client.applicationID(0) = Asc("M")
client.applicationID(1) = Asc("T")
client.applicationID(2) = CByte(0) ' "AA" 以上を設定
client.threadID = 50

' 構造体をパックされた列に変換
StructToRow clientrow.buf, ClientIDFldMap, client, LenB(client)

nStatus = BTRCALLID(BVERSION, _
                    0, _
                    versionBuffer, _
                    LenB(versionBuffer), _
                    sKeyBuffer, _
                    nKeyBufLen, _
                    0, _
                    client)

If nStatus = B_NO_ERROR Then

    ' パックされた列を構造体に変換
    RowToStruct versionBuffer.buf, Version_StructMap, versionstruct, _
        LenB(versionstruct)

    For i = 0 To 2
        If (versionstruct.ver(i).version > 0) Then
            msg = "Btrieve Versions returned are: " & _
                versionstruct.ver(i).version & "." & _
                versionstruct.ver(i).revision & "
```

```

        " " & versionstruct.ver(i).MKDEID
    PrintLB (msg)
End If
Next i

Else
    msg = "Btrieve B_VERSION status = " & nStatus
    PrintLB (msg)
End If

If nStatus = B_NO_ERROR Then

    ' Person テーブルを開く (sample.btr)
    nStatus = BTRCALL(BOPEN, _
        sPersonPosBlk, _
        PersonRow, _
        LenB(PersonRow), _
        ByVal sKeyBuffer, _
        nKeyBufLen, _
        nPersonKeyNum)

    msg = "Btrieve B_OPEN status = " & nStatus
    PrintLB (msg)
    If nStatus = B_NO_ERROR Then
        FileOpen = True
    End If
End If

If nStatus = B_NO_ERROR Then

    'GetEqual Btrieve 呼び出し
    lPersonID = 263512477 ' この社会保障番号のレコードを検索
    nStatus = BTRCALL(BGETEQUAL, _
        sPersonPosBlk, _
        PersonRow, _
        LenB(PersonRow), _
        lPersonID, _
        LenB(lPersonID), _
        nPersonKeyNum)

    msg = "Btrieve B_GETEQUAL status = " & nStatus
    PrintLB (msg)
    If nStatus = B_NO_ERROR Then

        ' 選択したレコードの出力

        PrintData PersonRow.buf

    End If
End If

```

Btrieve API の言語インターフェイス

```
' ファイルのステータスを取得
nStatus = BTRCALL(BSTAT, _
                 sPersonPosBlk, _
                 NewFileSpec, _
                 100, _
                 ByVal sKeyBuffer, _
                 nKeyBufLen, _
                 -1)

msg = "Btrieve B_STAT status = " & nStatus
PrintLB (msg)

If nStatus = B_NO_ERROR Then

    ' sample2.bt を作成して開く
    nStatus = BTRCALL(BCREATE, _
                    0, _
                    NewFileSpec, _
                    100, _
                    ByVal sKeyBuffer2, _
                    nKeyBufLen2, _
                    0)

    msg = "Btrieve B_CREATE status = " & nStatus
    PrintLB (msg)
End If
If nStatus = B_NO_ERROR Then
    nPersonKeyNum = 0
    nStatus = BTRCALL(BOPEN, _
                    sPersonPosBlk2, _
                    PersonRow, _
                    LenB(PersonRow), _
                    ByVal sKeyBuffer2, _
                    nKeyBufLen2, _
                    nPersonKeyNum)

    ' 元のファイルからデータを抽出し、新しいファイルへ挿入
    If nStatus = B_NO_ERROR Then
        File2Open = True
    End If
End If

If nStatus = B_NO_ERROR Then

    ' getFirst を実行してカレンシーを確立
    nPersonKeyNum = 2 'STATE-CITY index
    nStatus = BTRCALL(BGETFIRST, _
                    sPersonPosBlk, _
                    PersonRow, _
```

```

        LenB(PersonRow), _
        ByVal sKeyBuffer, _
        nKeyBufLen, _
        nPersonKeyNum)

    msg = "Btrieve B_GETFIRST status = " & nStatus
    PrintLB (msg)
End If

prebuffer.gneHeader.currencyConst = "UC"
While nStatus = B_NO_ERROR

    prebuffer.gneHeader.rejectCount = 0
    prebuffer.gneHeader.numberTerms = 2

    ' 最初の条件を代入
    prebuffer.term1.fieldType = 11
    prebuffer.term1.fieldLen = 3
    prebuffer.term1.fieldOffset = 108
    prebuffer.term1.comparisonCode = 1
    prebuffer.term1.connector = 2
    prebuffer.term1.value = "TX" & Chr(0)

    ' 2 つ目の条件を代入
    prebuffer.term2.fieldType = 11
    prebuffer.term2.fieldLen = 3
    prebuffer.term2.fieldOffset = 108
    prebuffer.term2.comparisonCode = 1
    prebuffer.term2.connector = 0
    prebuffer.term2.value = "CA" & Chr(0)

    ' プロジェクション ヘッダーを設定してレコード全体を読み込む
    prebuffer.retrieval.maxRecsToRetrieve = 20
    prebuffer.retrieval.noFieldsToRetrieve = 1

    prebuffer.recordRet.fieldLen = 157
    prebuffer.recordRet.fieldOffset = 0

    prebuffer.gneHeader.descriptionLen = Len(prebuffer)

    ' 定義済みの列から、パックされた配列をプリバッファーに作成
    StructToRow prebuffftype.buf, pregnebufferMap, prebuffer,
LenB(prebuffer)

    ' 定義済みの列から、パックされた配列をポストバッファーに作成
    StructToRow postbuffftype.buf, Post_GNE_BUFFERFieldMap, postbuffer, _
        LenB(postbuffer)

    ' プリバッファーをポストバッファー領域にコピー
    CopyMemory postbuffftype, prebuffftype, LenB(prebuffftype)

```

Btrieve API の言語インターフェイス

```
'GetNextExtended Btrieve 呼び出し
nStatus = BTRCALL(BGETNEXTEXTENDED, _
                 sPersonPosBlk, _
                 postbufftype, _
                 LenB(postbufftype), _
                 ByVal sKeyBuffer, _
                 nKeyBufLen, _
                 nPersonKeyNum)

msg = "Btrieve B_GETNEXTEXTENDED status = " & nStatus
PrintLB (msg)

'Get Next Extended は、ファイルの終わりまで到達してもレコードを
'返すことがあります
If ((nStatus = B_NO_ERROR) Or (nStatus = B_END_OF_FILE)) Then

    InsertNewData postbufftype.buf

End If

prebuffer.gneHeader.currencyConst = "EG"

Wend

nPersonKeyNum = 0
msg = " "
PrintLB (msg$)
If FileOpen = True Then

    '開いているファイルを閉じる
    nStatus = BTRCALL(BCLOSE, _
                    sPersonPosBlk, _
                    0, 0, 0, 0, 0)

    msg = "Btrieve B_CLOSE (sample.btr) status = " & nStatus
    PrintLB (msg)
End If

If File2Open = True Then
    nStatus = BTRCALL(BCLOSE, _
                    sPersonPosBlk2, _
                    0, 0, 0, 0, 0)

    msg = "Btrieve B_CLOSE (sample2.btr) status = " & nStatus
    PrintLB (msg)
End If

'リソースの解放
nStatus = BTRCALL(BRESET, _
```

```

        "", _
        0, _
        0, _
        CLng(0), _
        0, _
        0)

msg = "Btrieve B_RESET status = " & nStatus
PrintLB (msg)

End Sub
Private Sub Form_Load()
    InitFieldMaps
    txtInput.Text = "<path>%samples%sample.btr"
    txtOutput.Text = "<path>%samples%sample2.btr"

End Sub
' *****
'   ListBox に書き込むためのヘルパー関数
' *****
Sub PrintLB(Item As String)
    frmBtrv32.lstBtrv.AddItem Item
End Sub
' *****
'   このサブルーチンでは、最初のファイルからのデータを次のファイルに挿入します。
' *****
Private Sub InsertNewData(row() As Byte)
    Dim rec As POST_GNE_BUFFER
    Dim msg As String
    Dim i As Integer
    Dim personRecord As PersonRecType
    Dim personRec As PersonRowType
    Dim DataLen As Integer
    Dim nStatus As Integer '

    ' パックされた列を構造体に変換
    RowToStruct row, Post_GNE_BUFFERfieldMap, rec, LenB(rec)
    msg = "GetNextExtended returned " & rec.numReturned & " record(s)."
    PrintLB (msg)

    For i = 0 To rec.numReturned - 1
        personRecord = rec.recs(i).personRecord
        StructToRow personRec.buf, PersonFldMap, personRecord,
LenB(personRecord)
        nStatus = BTRCALL(BINSERT, _
            sPersonPosBlk2, _
            personRec, _
            LenB(personRec), _
            ByVal sKeyBuffer2, _
            nKeyBufLen2, _

```

Btrieve API の言語インターフェイス

-1) ' カレンシー変更なし

```
If (nStatus <> B_NO_ERROR) Then
    msg = "Btrieve B_INSERT status = " & nStatus
    PrintLB (msg)
End If
Next i

msg = "Inserted " & rec.numReturned &
      " records in new file, status = " & nStatus
PrintLB (msg)

End Sub
'*****
' このサブルーチンでは、選択したレコードのデータを出力します。
'*****
Private Sub PrintData(row() As Byte)
Dim rec As PersonRecType
Dim msg As String

    ' パックされた列を構造体に変換
RowToStruct row, PersonFldMap, rec, LenB(rec)
msg = " "
PrintLB (msg$)
msg = "Selected fields from the retrieved record are: "
PrintLB (msg)
msg = "ID =          " & Chr$(9) & rec.ID
PrintLB (msg)
msg = "First Name = " & Chr$(9) & rec.FirstName
PrintLB (msg)
msg = "Last Name =  " & Chr$(9) & rec.LastName
PrintLB (msg)
msg = "Address =    " & Chr$(9) & rec.Street
PrintLB (msg)
msg = "City =       " & Chr$(9) & rec.City
PrintLB (msg)
msg = "State =      " & Chr$(9) & rec.State
PrintLB (msg)
msg = "Country =    " & Chr$(9) & rec.Country
PrintLB (msg)
msg = "Zip =        " & Chr$(9) & rec.Zip
PrintLB (msg)
msg = "Phone =     " & Chr$(9) & rec.Phone
PrintLB (msg)
msg = " "
PrintLB (msg$)

End Sub
```

サンプルプログラムのコンパイル、リンク、実行

➤ Visual Basic では、32 ビット サンプルプログラムのコンパイル、リンク、実行を、以下の方法で行います。

- 1 Visual Basic プログラミング環境では、[ファイル] メニューの [プロジェクトを開く] を選択します。
- 2 Btr3VbSample.vbp プロジェクト ファイルを開きます。
- 3 入出力テキスト ボックスを編集し、使用する sample.btr および sample2.btr のフルパスを指定します。
- 4 ツールバーの [開始] ボタンをクリックします。
サンプルプログラムがコンパイルおよびリンクされ、[Btrieve Visual Basic Sample] ウィンドウが作成されます。
- 5 [Btrieve Visual Basic Sample] ウィンドウの [Run Test] ボタンをクリックします。
サンプルプログラムが実行されます。

索引

A

Access のバージョン	5
ActiveX	
VB プロジェクトへのインターフェイス追加	
89	
ActiveX Data Object (ADO)	33
ActiveX Visual Basic	
ActiveX インターフェイスを準備する ..	64
ActiveX インターフェイス	
Delphi IDE	40
Pervasive PSQL アプリケーション作成に使用	87
Pervasive PSQL ファイルへ接続	92
ADO	24, 26
ALTER TABLE ステートメントの例	19

B

Btrieve API	
VB プロジェクトでの DLL の参照	72
Visual Basic	
ファイルを閉じる	75
Visual Basic でのグローバル構造体の初期化	
72	
ファイルの作成 , Visual Basic	76
ファイルを閉じる , Visual Basic	76
呼び出し	21
レコードの更新 , Visual Basic	78
レコードの挿入 , Visual Basic	78

C

C++ Builder	134
COBOL	
言語インターフェイス	146
ColumnWidth プロパティ	98

D

DDF	8, 23
DdfPath プロパティ	93
Delphi	
2 つの VAccess コントロールを結合する	47

Btrieve API	49
Get オペレーションの実行	55
IDE に ActiveX インターフェイスをインストールする	40
ODBC データベースへの接続	57
SQL を使用してクエリを実行する , ODBC	
57	
SQL を使用して更新する , ODBC	58
SQL を使用して挿入する , ODBC	59
Step オペレーションの実行	54
VAccess コントロールの使用	41
VAList ボックス コントロールの使用 ..	44
言語インターフェイス	150
コントロールのバインド , ODBC	58
セグメント インデックスの使用	55
データの検索	56
パラメータ Update ステートメントを使用する , ODBC	60
パラメーター Insert ステートメントを使用する , ODBC	60
パラメーター クエリを使用する , ODBC	
59	
ファイルの作成	51
ファイルを閉じる	50
ファイルを開く	49
レコードの更新	53
レコードの削除	54
レコードの挿入	53

F

FIELD.DDF ファイル	8, 23
FILE.DDF ファイル	8, 23

G

GetFirst メソッド	95
---------------------	----

I

INDEX.DDF ファイル	8, 23
IndexNumber プロパティ	94
INSERT ステートメントの例	19

J

Jet エンジンの制約 12

L

Location

プロパティ 93

Login サンプル , Btrieve API 123

O

ODBC

API 呼び出し 21

Microsoft Access での Pervasive PSQL の使用
5

MSRDC を使用して更新する , Visual Basic
83

MSRDC を使用して挿入する , Visual Basic
81

SQL を使用してクエリを実行する , Delphi
57

SQL を使用して更新する , Delphi 58

SQL を使用して更新する , Visual Basic . 82

SQL を使用して選択する , Visual Basic . 79

SQL を使用して挿入する , Delphi 59

SQL を使用して挿入する , Visual Basic . 81

アドミニストレーター 7, 23

コントロールのバインド , Delphi 58

接続文字列 17

データベースへの接続 , Delphi 57

パラメーター Insert ステートメントを使用す
る , Delphi 60

パラメーター Insert ステートメントを使用す
る , Visual Basic 83

パラメーター Update ステートメントを使用
する , Delphi 60

パラメーター Update ステートメントを使用
する , Visual Basic 84

パラメーター クエリを使用する , Delphi 59

パラメーター クエリを使用する , Visual
Basic 83

ODBC の接続文字列 17

P

Pervasive PSQL アプリケーション

作成 87

R

[Returns Records] フィールド 19

S

SELECT ステートメントの例 17

SQL Data Manager 24

SQL Data Manager ユーティリティ 8

SQL パススルー クエリ

アクセス 16

SQL ステートメントの引用符 18

T

TableName プロパティ 93

U

UPDATE ステートメントの例 19

V

VAccessName プロパティ 96

VAccess コントロール 90

Pervasive PSQL ファイルへ接続 92

Visual Basic 64

Visual Basic を使用した 2 つのコントロール
の結合 69

アプリケーション開発に使用 90

結合 106

VAccess コントロールの結合 106

VACommandButton 102

VACommand コントロール

Visual Basic を使用した Btrieve オペレーショ
ンの実行 70

VA DdfPath プロパティ 93

VAList ボックス コントロール

Delphi 44

Visual Basic 67

VARecordList プロパティ 97

VAText コントロール

Visual Basic 66

VAText ボックス 96, 102

VAText ボックス コントロール 96

VAVScrollBar 99

Visual Basic

2 つの VAccess コントロールの結合 69

ActiveX インターフェイスを準備する .. 64

Btrieve API 72

ファイルを開じる.....	75	VAccess.....	90
Btrieve API とグローバル構造体.....	72	VAccess, アプリケーション開発に使用.....	90
MSRDC を使用して更新する, ODBC....	83	VAText ボックス.....	96
MSRDC を使用して挿入する, ODBC....	81	結合.....	106
SQL を使用して更新する, ODBC.....	82		
SQL を使用して選択する, ODBC.....	79	さ	
SQL を使用して挿入する, ODBC.....	81	サードパーティ製データコントロール....	21
VAccess コントロールの使用.....	64	サーバーエンジン.....	2
VACommand コントロールを使用して		作成	
Btrieve オペレーションを実行する....	70	Pervasive PSQL アプリケーション.....	87
VAList ボックス コントロールの使用...	67	データブラウザー.....	89
VAText コントロールを使用する.....	66	サンプルデータベースの場所.....	6, 22
VB プロジェクトへの ActiveX インターフェ			
イスの追加.....	89	す	
言語インターフェイス.....	176	スタンドアロンエンジン.....	2
バージョン.....	21	ステータスコード	
パラメーター Insert ステートメントを使用		501.....	17, 18
する, ODBC.....	83	せ	
パラメーター Update ステートメントを使用		セキュリティ	
する, ODBC.....	84	Btrieve Login サンプル.....	123
パラメーター クエリを使用する, ODBC 83		設定	
ファイルの作成, Btrieve API.....	76	IndexNumber.....	94
ファイルを開じる, Btrieve API.....	76		
ファイルを開く, Btrieve API.....	74	つ	
レコードの更新, Btrieve API.....	78	通信.....	2
レコードの挿入, Btrieve API.....	78		
		て	
あ		データコントロールのバインド.....	26, 49
アドミニストレーター		データ辞書ファイル.....	8, 23
ODBC.....	7, 23	データソース.....	7, 23
アプリケーション		データバウンド	
Pervasive PSQL の作成.....	87	グリッドコントロール.....	24
		リストコントロール.....	24
い		データブラウザー	
インターフェイス		作成.....	89
ActiveX.....	92	テーブル	
		インポート.....	13
く		リンク.....	10
クエリ		テーブルのインポート.....	13
アクセス.....	16	テーブルのリンク.....	10
クライアント / サーバー エンジン.....	2	デモデータの場所.....	6, 22
こ			
構文エラー.....	17, 18		
コントロール			

ね

ネットワーク環境..... 2

は

はじめに..... 1

ふ

プログラム

Pervasive PSQL の作成..... 87

プロパティ

IndexNumber..... 94

Location..... 93

TableName..... 93

り

リクエスター..... 2

ろ

ログイン

Btrieve オペレーション..... 123