4D Server

リファレンス

Windows® and Mac™OS

ACI

4D Server リファレンス

Windows® and Mac™ OS

このマニュアルに記載されている事項は、将来予告なしに変更されることがあり、いかなる変更に関してもACI SAは一切の責任を負いかねます。このマニュアルで説明されるソフトウェアは、本製品に同梱のLicense　Agreement（使用許諾契約書）のもとでのみ使用することができます。

ソフトウェアおよびマニュアルの一部または全部を、ライセンス保持者がこの契約条件を許諾した上での個人使用目的以外に、いかなる目的であれ、電子的、機械的、またどのような形であっても、無断で複製、配布することはできません。

4th Dimension、4D Server、4D、4Dロゴ、ACIロゴ、およびその他のACI製品の名称は、ACI SAの商標または登録商標です。

Microsoft とWindows はMicrosoft Corporation 社の登録商標です。

Apple, Macintosh, Mac, Power Macintosh, Laser Writer, Image Writer, ResEdit, QuickTimeはApple Computer Inc.の登録商標または商標です。

その他、記載されている会社名，製品名は、各社の登録商標または商標です。

注意

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

目次

概要

4D Serverは、クロスプラットフォームでマルチユーザに対応する4th Dimension用データサーバおよびアプリケーションサーバです。

4D Serverでは、クライアント／サーバアーキテクチャに基づいた、マルチユーザデータベースおよびカスタムアプリケーションを作成し、使用することができます。プラットフォームに依存しないクライアント／サーバアーキテクチャにより、WindowsおよびMacintosh上の4D Clientからデータベースをシームレスに管理できます。4D Serverには大規模システムに向けて、プロフェッショナル仕様の開発用ツール、拡張性、データ保護機能、コネクティビティオプションが用意されています。

4D Serverのアーキテクチャは、完全に統合されており、クライアントとサーバの両方で1つの4Dアプリケーションを使用します。4D Serverにより、開発者はフロントエンドアプリケーションとバックエンドアプリケーションを個別に設計する手間から解放されます。それだけではありません。4D Serverは“管理不要”のサーバです。インストールや使用、管理が容易でありながらも、非常にコストエフェクティブなサーバです。

ローエンドのファイルシェアリングベースのシステムと複雑なSQLベースのRDBMSとの中間に位置するもの、それが4D Serverです。4Dコネクティビティプラグインを利用すれば、4D Serverアプリケーションを既存の大規模データベース（Oracle、Sybase、ODBC対応サーバ等）とスムーズに連携させることができます。4D Serverは、あらゆる規模の企業のワークグループから派生するニーズに対応します。

統合されたバックエンドアーキテクチャとフロントエンドアーキテクチャ

4D Serverでは、フロントエンドアプリケーションとバックエンドアプリケーションは同じものです。クライアントソフトウェアとサーバアプリケーションは、4th Dimensionという1つの製品が2つの役割を果たしています。4D Serverアプリケーション自体は、4D Serverと4D Clientという2つの部分に分かれており、それぞれがクライアント／サーバアーキテクチャにおける構成要素となります。

4D Server部分は、サーバマシンに常駐し、サーバ上のデータベースを保存し、管理します。エンドユーザは自分自身が使用しているマシン（クライアント）からこのデータベースを利用します。

各クライアントマシンには、4D Clientのコピーが常駐します。ユーザは4D Clientを使用してサーバ上のデータベースにアクセスし、データの追加、レポートの作成、データベースデザインの変更等のデータベース作業を行います。4th Dimensionで行う作業はすべて、4D Serverと4D Clientを使用して行うことができます。

クライアント／サーバ環境で操作するために、他のミドルウェアや開発は不要です。4D、4D Client、4D Serverでは、同一のインタフェースツール、言語、情報管理システムが使用されます。

1つのプラットフォーム（Windows 95、Windows NT、Power Macintoshのいずれか）用に作成されたシングルユーザ用アプリケーションはすべて、ワークグループ対応のクライアント／サーバソリューションへと簡単に拡張できます。また逆に、4D Serverで作成されたアプリケーションを自動的にスタンドアロンアプリケーションに変えることも可能です。

“管理不要”のデータサーバとアプリケーションサーバ

4D Serverは、ユーザに重点を置いた4Dの特長を受け継ぎました。この結果、4D Serverは完全なプラグアンドプレイシステム（PNP）となっています。

4D Serverのメインウインドウ

- **わかりやすくグラフィカルなオンライン管理ウインドウ：**4D Serverのメインウインドウには、重要な情報が自動的に表示されます。4D Serverに割り当てられているメモリ合計、データキャッシュ、接続中のユーザ数と名前、プロセス数とステータス、サーバの稼働状況、キャッシュ率等の情報です。
- **システム自動構成とスケーラビリティ：**4D Serverは、システムの再構成や再設計を行わなくても、新しいプロトコル、クライアント、プラグインを追加できるように設計されています。
- **クライアントワークステーションに対するダイナミックな自動更新とバージョン管理：**データベースの変更や、プラグインの追加、削除、修正があった場合には、すべての4Dクライアントがダイナミックに自動更新されます。
- **ローカルやリモートのプロトコル混在環境へのへの非同期自動接続：**4D Serverには統合された仮想ネットワークマネージャが装備され、複数プロトコルのインストールや同時利用も簡単です。4D Serverと4D Clientは、物理的なネットワーク層に関わらず、TCP/IP、Novell社のIPX/SPX、AppleTalkといったネットワークプロトコルを個別または同時にサポートします。
- **4D ClientとWeb接続のセッションおよび状況の自動管理：**4D Serverは、テーブル／プロセス／ユーザの組み合わせごとに現在の作業環境を自動的に作成し、管理します。このセッションベースのアーキテクチャにより、ユーザごとの各プロセスでは、データを個別かつ同時に操作することができます。
- **自動レコードロック機能：**4D Serverでは、レコードのロックや解放が自動的に行われ、“使用中”のレコードの変更に関わるトラブルを回避することができます。また、レコードロック機能により、ページロックやファイルロックに関連する問題も解消されます。
- **統合ユーザインタフェースメッセージシステム：**デスクトップの領域から誕生した4D Serverは、最新の統合開発環境で提供されているユーザインタフェースをすべて備えています。例えば、接続解除やバックアップ等をスケジュール管理する等、管理側の作業をクライアントに通知することができます。
- **自動起動、自動終了メソッド：**4D Serverでは、特定のイベントに応じて5つのサーバデータベースメソッドが自動的に起動されます。この5つのメソッドとは、「On Server Startup」、「On Server Shutdown」、「On Server Open Connection」、「On Server Close Connection」、「On Web Connection」です。例えば、「On Server Startup」メソッドは、セッション全般で必要となるオブジェクトを自動的に初期化し、ロードします。

機能

4D Serverには、4th Dimensionの機能に加えて次の機能があります。

- **マルチユーザデータ管理：**複数のユーザがデータベース作業を同時に実行できます。複数のユーザが同じファイルあるいは異なるファイルのレコードを追加、変更、削除、検索、ソート、印刷することが可能です。データの整合性は、内蔵のレコードロックシステムによって保持されます。

- **マルチユーザ開発：**複数のユーザがデータベースの開発や設計を同時に行えます。例えば、同時に複数ユーザがファイル定義の編集、およびレイアウト、スクリプト、メソッドの作成、変更することができます。データベースデザインの整合性は、内蔵のオブジェクトロックシステムによって保持されます。

- **プラットフォームに依存しないクライアント／サーバアーキテクチャ：**このアーキテクチャにより、MacintoshおよびWindows上の4D Clientに対するデータベースパフォーマンスはシームレスに管理されます。また、異なるプラットフォーム上での同時マルチ開発や、異種混合ハードウェア環境で動作する4D Clientにより入力、変更されるすべてのデータのインタフェースも管理されます。

- **WindowsおよびMacOSベースの4Dプラグインアーキテクチャ：**Windows版4D Serverでは、サーバマシンにWin4DXおよびMac4DXフォルダにWindowsベースおよびMacOSベースの4Dプラグインをインストールできます。このアーキテクチャにより、プラットフォームに依存しない4Dプラグインの配布を簡単に行えます。クライアントのプラットフォームが何であれ、4D Serverおよび4D Clientによりプラグインはスムーズに処理されます。

- **ビルトインWebサーバ：**4th Dimensionと同様に、4D ServerにはWeb Serverエンジンが組み込まれており、Web上に4Dデータベースを公開することができます。データベースはWeb上に直接公開されるので、データベースシステム、Webサイト、この両者の間のCGIインタフェースを開発する必要はありません。データベースがWebサイトなのです。ビルトイン4D Web Serverの詳細については『4th Dimensionランゲージリファレンス』マニュアルの第61章「Webサーバコマンド」の「Webサービス：概要」の項を参照してください。

- **トリガ：**トリガは、テーブルに付随するメソッドであり、テーブルのプロパティです。トリガはコールする必要はありません。ユーザがテーブルレコードを操作（追加、削除、変更、ロード）するたびに、4Dデータベースエンジンにより自動的に起動されます。非常に簡単なトリガを作成し、後で更に洗練されたものにしていくことができます。トリガはデータベース上のレコードに対する「不正な」操作を防止することができます。トリガは、データを偶然失ったり、不正に変更することを避けるだけでなく、テーブル操作を制限できる非常に強力なツールです。例えば、請求書管理システムでは、請求する顧客を指定しなくては請求書を追加できないようにすることができます。4D Serverでは、トリガはサーバマシン上で実行されます。4D Clientまたは4D Openベースのアプリケーションのいずれでも、すべてのクライアントは、トリガにより実行されるデータベースルールに従う必要があります。4Dトリガの詳細については『4th Dimension ランゲージリファレンス』マニュアルの第57章「トリガコマンド」を参照してください。

- **ストアードプロシージャ：**4Dメソッドは、独立したプロセスとしてサーバマシン上でローカルに実行するよう作成することができます。クライアント／サーバの分野の用語では、この新しい機能は“ストアードプロシージャ”と呼ばれます。しかし、4D Serverはストアードプロシージャの通常の概念の更に上を行くアーキテクチャを提供します。4D Serverでは、ストアードプロシージャは実際にはカスタムサーバプロセスであり、サーバマシンまたはクライアントマシン上で実行される他のすべてのプロセスからは独立して、非同期にコードを実行します。通常のクライアント／サーバアーキテクチャでは、ストアードプロシージャは実行され、結果を戻します（同期的または非同期的に）。4D Serverの場合は、ストアードプロシージャを開始し、クライアント／サーバセッションを通して、要求に応じてクライアントから送られるメッセージに応答することができます。同時に、クライアントとはまったくやり取りを行わないストアードプロシージャを実行することもできますが、その代わりに4Dコネクティビティプラグインまたは4D Openを使用して、SQLベースのサーバや他の4D Serverとデータの同期をとります。同時実行できるストアードプロシージャの数には制限はありません（ハードウェアやメモリの制限は除く）。4D Serverのストアードプロシージャは独自のプロセス内で実行され、そのため他のユーザプロセスと同様に、独自のデータベースコンテキスト（カレントセレクション等）が維持されます。さらに、4D言語では、クライアントプロセスがストアードプロシージャのプロセス変数（BLOB変数等）を読み書きできるコマンドが提供されます。これらのコマンドによって、クライアントとストアードプロシージャ間で、洗練された、柔軟な通信を行うことができます。事実、ストアードプロシージャを使用すると、4D Serverに新しいサービスやカスタムサービスを追加できます。詳細については第7章「ストアードプロシージャ」を参照してください。

- **サーバパス：**「パスワードアクセス」エディタを使って、ユーザのパスワードと共にサーバデータベースへのパスを保存することができます。この機能により、ユーザは4D Clientの“パスドキュメント”アイコンをクリックするだけで、サーバ上のデータベースに接続することが可能です。ゾーンが設定されているネットワーク上で使用すると、4D Clientはこのパスを使って正しいデータベースに自動的にアクセスします。

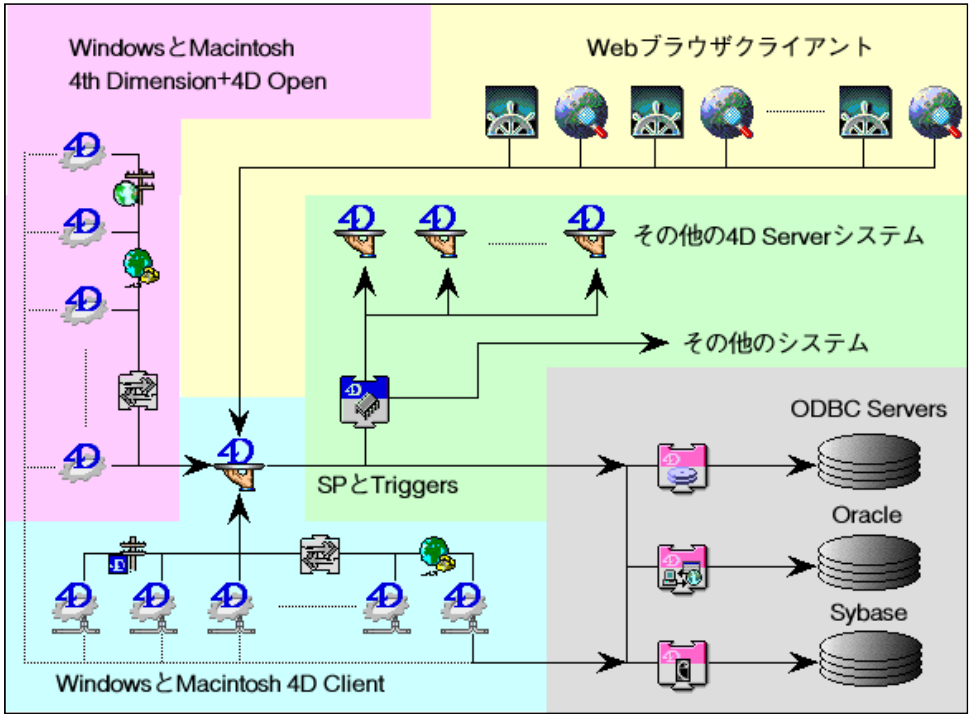
- **4D Open：**4D Server用のAPI（アプリケーションプログラムインタフェース）である4D Openを使用すると、ユーザは、4D Client以外のプログラムから4D Serverデータベースに接続できます。これらのプログラムは、Windows、Macintoshのいずれのアプリケーションでもかまいません。4D Openは、4th Dimensionおよび4D Clientに組み込むことができます。ユーザは複数のサーバに同時に接続できるので、部署別または部署間での分散型システムが実現します。最後に、トリガやストアードプロシージャ（つまり、サーバマシン上で実行されるメソッド）内で4D Openを使用すると、データの自動複製や自動配布を行うために、4D Serverが他の4D Serverに接続するようなシステムを作成できます。つまり、4D Openにより、4D Serverが自分自身のアーキテクチャのクライアントになります。

- **4D Backup：**4D Backupはデータベースのバックアップコピーを作成します。ディスク上に単純なバックアップを作成することから、オリジナルデータベースの複製となるミラーデータベースを管理することまで、さまざまな方法を選択できます。4D Backupをデータベースにインストールすると、4th Dimensionのモジュールとして統合

されます。4D Serverを使用している場合には、サーバコンピュータ上の4D Serverから4D Backupモジュールを開くこともできます。

- **コネクティビティプラグイン：**ACIコネクティビティプラグイン（4D ODBC、4D for ORACLE、4D SQL Server）を使用すると、4D Serverおよび4D Clientは、ORACLE、Sybase SQL Server、およびその他のODBCデータベースサーバ等の、メインフレームやミニコンピュータのデータベースに直接アクセスできます。これらのデータベース間では、情報をインタラクティブに共有できます。さらに、現在ACIでは、ODBCクライアントが4D Serverに接続し、作業できるような4D Server ODBCドライバを開発中です。
- **リモート接続：**モデム経由で通常の電話回線またはデジタル電話回線から4D Serverデータへアクセスすることができます。Macintoshでは、Apple Remote Access（ARA）経由で接続をサポートします。詳細については、ARAのマニュアルを参照してください。Windowsでは、4D ServerはRemote Access Service (RAS) 経由で接続をサポートします。詳細については、Windows NTやWindows 95のマニュアルを参照してください。

次の図は、4D Serverのデータベースシステムをまとめたものです。



4D Server (WindowsまたはMacintosh)

4D Client (WindowsまたはMacintosh)

4th Dimension + **4D Open** (WindowsまたはMacintosh)

ACI Connectivity plug-ins

Network connections (TCP/IP, Novell's IPX/SPX, AppleTalk)

Apple Remote Access

Remote Access Windows connections

4D Serverアーキテクチャ

クライアント／サーバアーキテクチャを使用すると、4D Serverは単にデータベースの保存と管理を行うだけではなく、クライアントに対するサービスも提供します。これらのサービスはネットワークを介した要求と応答のシステムによって処理されます。

例えば、一連のレコードを検索する場合に、クライアントマシンはサーバへ検索要求を送ります。サーバはこの要求を受け取ると、サーバマシン上で検索処理を実行し、完了するとその結果（検索レコード）を返します。

4D Serverのアーキテクチャはクライアント／サーバモデルに基づいています。ここ数年の間に、クライアント／サーバアーキテクチャは、旧式のファイルシェアリングアーキテクチャをしのぎ、マルチユーザデータベースとしては最も効率的なモデルになっています。

4D Serverが採用しているクライアント／サーバアーキテクチャは、ミニコンピュータの世界で使用されているものと似ています。しかし、4D Serverには次に示す2つの重要な新しい機能があります。

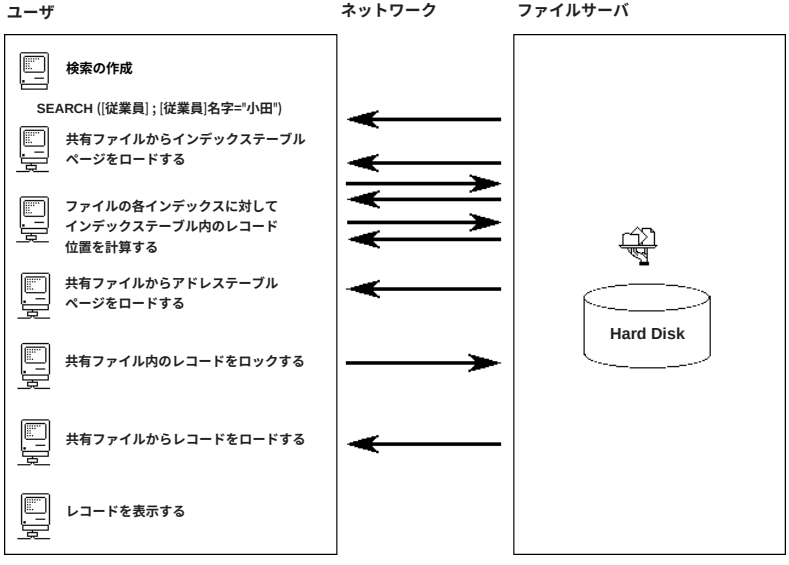
- データベースのすべてのレベルで使用できる、親しみやすいグラフィカルインタフェース
- より効率が良く、高速な統合アーキテクチャ

ファイルシェアリングアーキテクチャ

クライアント／サーバアーキテクチャが導入される以前、マルチユーザシステムには、ネットワークアーキテクチャのファイルシェアリングモデルが使用されていました。ファイルシェアリングモデルでは、すべてのユーザは同一のデータを共有しますが、データ管理は中央のデータベースエンジンによって制御されていません。各クライアントマシンにはデータベースストラクチャとエンジンのコピーを格納する必要があり、一方でサーバにはネットワーク上でファイルを共有するために必要なソフトウェアしかありません。

ファイルシェアリングモデルのもとでは、各ワークステーションがすべてのデータ変更をローカル上で行います。処理ごとに多量の更新を必要とし、ネットワーク付加の増大につながりました。

次の図は、名字が“ 小田 ”であるすべての人をデータベースから検索する場合のネットワークの使用量を示しています。



ファイルシェアリングモデルのもう1つの欠点は、メモリキャッシュを利用してメモリ上にレコードを保持できないということです。ファイルシェアリングモデルでレコードがメモリ内に保持されると、各ユーザが同じレコードを異なるバージョンでキャッシュに格納する可能性があり、データに矛盾が生じてしまうためです。したがって、ユーザはレコードにアクセスするたび、ファイルサーバからレコードをダウンロードする必要があります。これはネットワーク負荷を増大させ、レコードアクセス時間の増加につながります。

異種混合クライアント／サーバアーキテクチャ

クライアント／サーバアーキテクチャは効率が良く、高速なので、ミニコンピュータの世界では大規模データベースシステムで広範囲に使用されています。このアーキテクチャでは、パフォーマンスを向上させるために、サーバマシンとクライアントマシンが作業を分担します。

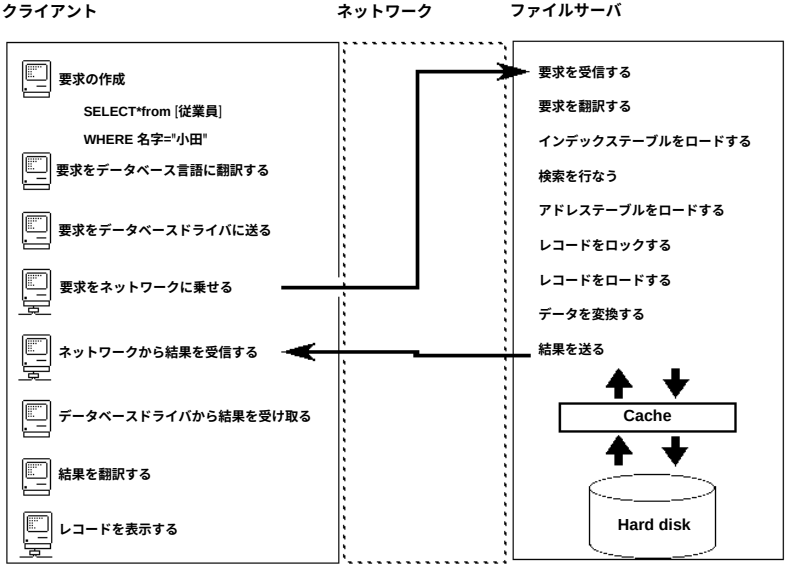
サーバには中心となるデータベースエンジンがあり、データを格納し、管理します。データベースエンジンは、ディスク上に格納されたデータをアクセスする唯一のソフトウェアです。クライアントがサーバに要求を送ると、サーバは結果を返します。結果はクライアントが変更する特定のレコードであったり、ソートした一連のレコードの場合もあります。

一般に、ほとんどのクライアント／サーバアーキテクチャは、異種混合アーキテクチャと呼ばれていますが、これはクライアントマシン上で実行しているフロントエンドアプ

リケーションとサーバマシン上で実行しているデータベースエンジンに別々の製品が使用されるためです。このような場合には、クライアントとサーバの間に入って翻訳を行うデータベースドライバが必要です。

例えば、レコードを検索する場合、クライアントはサーバに検索要求を送ります。データベースはサーバ上に格納されているので、サーバはサーバマシン上でローカルにコマンドを実行し、結果をクライアントに返します。

次の図は、名字が“小田”であるすべての人をデータベースから検索し、見つかった最初のレコードを表示するようにサーバに要求した場合のネットワークの使用量を示します。



この例により、クライアント／サーバアーキテクチャとファイルシェアリングアーキテクチャでは2つの点で大きく異なることがわかります。

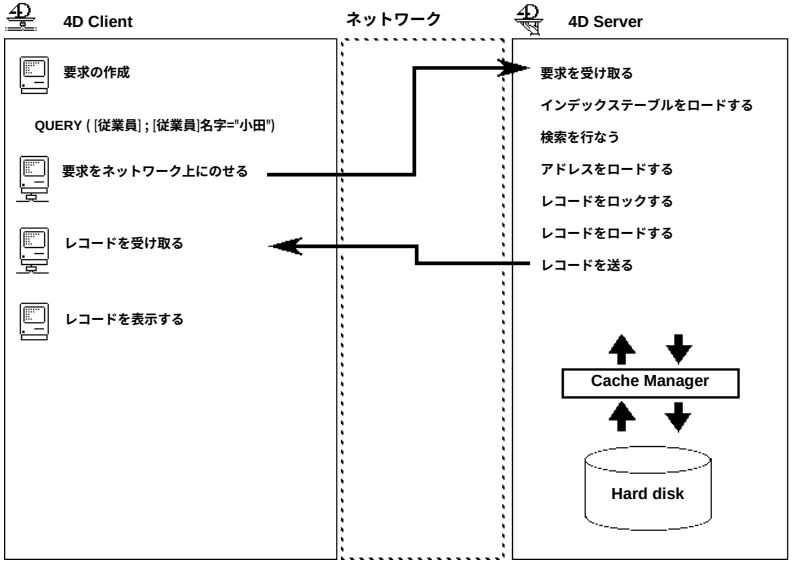
- **クライアント／サーバアーキテクチャではキャッシュを使用できる：**データに物理的なアクセスを行うのはエンジンだけのため、サーバはディスクに書き込まれるまで、変更レコードを保持するためのキャッシュをメモリ上に持つことができます。データを送りだすところは1カ所なので、クライアントは必ず最新版レコードを受け取ることができます。中央のキャッシュメカニズムを使用することにより、データの整合性を保証すると共に、ディスクにアクセスするのではなく、メモリにアクセスするため、データベース処理速度の向上にもつながります。ファイルシェアリングモデルでは、アクセスはすべてディスクアクセスです。
- **低レベルのデータベース処理はサーバ上で実行される：**クライアント／サーバアーキテクチャでは、インデックスやアドレステーブルのブラウジング等、低レベルのデータベース処理はサーバマシンのスピードに合わせてサーバマシン上でローカルに実行

されるので、処理速度が大幅に速くなります。ファイルシェアリングアーキテクチャでは、同じ処理を行っても、ネットワークの通信速度とクライアントマシンの制約のため遅くなります。

4D Serverの統合クライアント／サーバアーキテクチャ

ほとんどのクライアント／サーバアーキテクチャでは、クライアントソフトウェアとサーバソフトウェアは2つの異なる製品であり、互いに“話し合う”ためにはコミニュケーションレイヤが必要です。4D Serverでは、クライアント／サーバアーキテクチャは完全に統合されています。4D Serverと4D Clientは、同一の構造を共有し、直接通信を行うアプリケーションです。

4D Serverと4D Clientは同じ言語を使用するので、問い合わせ言語を翻訳する必要がありません。クライアントとサーバの作業の分担は、透過的であり、4D Serverが自動的に管理します。
作業の分担は、1つの要求が1つの応答を返す形で構成されています。以下に示すように、クライアントには次の役割があります。



■ **要求：**4D Clientは4D Serverに要求を送ります。これらの要求はクエリエディタや並べ替えエディタ等の組み込みエディタ、またはプロシージャ言語を使用して行います。4D Clientには、メソッドを作成し、変更できるエディタが用意されています。また、変数や配列等のメソッド要素も管理されます。

- **応答の受信：**4D Clientは4D Serverからの応答を受け取り、ユーザインタフェース（フォームに様々なレコードが表示する等）を介してユーザに最新情報を示します。例えば、名字が“田中”であるすべてのレコードをクライアントが要求した場合に、4D Clientは4D Serverから結果のレコードを受け取り、それをフォームに表示します。

サーバには次の役割があります。

- **スケジューリング：**4D Serverは、同時接続およびクライアントが作成した同時プロセスをすべてスケジュールするためのマルチタスキングアーキテクチャを使用します。
- **ストラクチャオブジェクトとデータオブジェクト：**4D Serverはフィールド、レコード、フォーム、メソッド、メニュー、リスト等、すべてのデータオブジェクトとストラクチャオブジェクトを格納し、管理します。
- **キャッシュ：**4D Serverは、レコードの他にセレクションやセット等の特定のクライアントに固有のデータオブジェクトを納めるキャッシュを維持します。
- **低レベルデータベース処理：**4D Serverは、クエリやソート等、インデックステーブルやアドレステーブルを使用する低レベルのデータベース処理を実行します。

この作業の分担は、4D Serverと4D Clientが独自の形式で統合されているので、非常に効率良く行われます。4D Serverのアーキテクチャの統合は、すべてのレベルに及んでいます。

- **要求レベル：**4D Clientが4D Serverにクエリやソート等の要求を送信すると、4D Clientは4D Serverと同じ内部構造を使用してクエリ処理やソート処理の記述を送信します。
- **ストラクチャレベルまたはデータレベル：**4D Clientと4D Serverがデータやストラクチャのオブジェクトをやり取りする場合、どちらのアプリケーションも同じ内部形式を使用します。例えば、4D Clientでレコードが必要な場合には、4D Serverはディスクやメモリキャッシュに入っていた場合の形でデータを送信します。同様に、4D Clientがレコードを更新する場合には、4D Clientが4D Serverへデータを送信し、4D Serverは受信したそのままのデータをキャッシュに格納します。
- **ユーザインタフェースレベル：**4D Clientがレコードのリストを表示する場合、レコードを表示するために使用されるフォームは、クライアント／サーバアーキテクチャの役割を果たしています。例えば、次の図は、[従業員]テーブルを検索した結果を示しています。


従業員：256 / 4096

名字	郵便番号	都道府県	住所
柴田	170	東京都	豊島区南大塚1234-5678
嶋田	101	東京都	千代田区岩本町33-33-33
原田	530	大阪府	大阪市北区9-9-9
深山	210	神奈川県	川崎市東川崎1-0-0
武藤	390	長野県	松本市松本1-2-3
刀根	209	東京都	府中市府中10番地
藤上	810	福岡県	福岡市中央区天神7丁目
佐藤	058	北海道	札幌市札幌55-55-55
魚住	862	熊本県	熊本市1丁目2番地
後藤	227	神奈川県	横浜市緑区3丁目
高崎	435	静岡県	浜松市浜松町12-34
安永	820	福岡県	飯塚市新飯塚7番地

上記のウインドウは、一度に4フィールドずつ12レコードしか表示できないため、4D Serverは12レコードだけを送信します。レコード全体を送る代わりに、4D Serverはウインドウに表示できるだけの数のレコードとフィールドを送ります。ユーザがフォームをスクロールした場合には、必要に応じて4D Serverから残りのレコードやフィールドが送信されます。この最適化により、レコードやフィールドは必要な場合にだけ送られ、ネットワーク使用量が削減されます。

インストール環境のチェック

「10分間で知る4D Server」の章は、以下の方法の概略を説明するチュートリアルです。

- サーバデータベースを作成する
- 作成したサーバデータベースにクライアントを接続する
- テーブル、フィールド、フォーム、メニュー、メソッド等のデータベースストラクチャを作成する
- 2番目のユーザを接続し、同時に作業をする
- Webブラウザと接続する

これには、以下の環境が必要です。

- 4D Serverがインストールされているマシン
- 4D Clientがインストールされている2台のマシン
- Webブラウザがインストールされている1台のマシン

4D Serverと4D Clientでの作業を初めて行う場合は、その前にインストール環境を確認することをお勧めします。確認するためには、この節を参照してください。

参考：この節は、バージョン6.5.2のリリース時に作成されたものです。したがって、たとえばWindowsでは、4D Serverアプリケーションは4D Server6.5.2フォルダに入っているものとして説明されています。これ以降のバージョンを使用している場合、フォルダ名は4D Server6.5.2や4D Client**6.5.2のようにそのバージョンを反映します。**

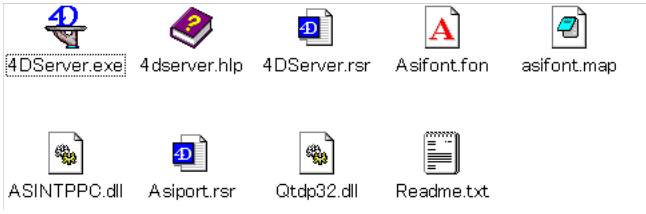
Windowsの場合

インストールプログラムで示されたデフォルトフォルダに4D Serverと4D Clientをインストールした場合には、次に示すフォルダとファイルがディスク上にあるはずです。

4D Server

ACI\Programsディレクトリに、**4D Server6.5.2**ディレクトリがインストールされます。

4D Server6.5.2フォルダには、4D Serverアプリケーションと関連ファイルがあります。

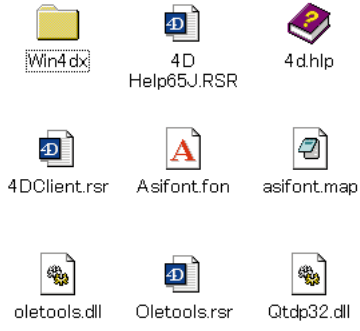


4D Serverを起動するには、“ 4DSERVER.EXE ”アイコンをダブルクリックします。

4D Client

ACI\Programsフォルダに、4D Client**6.5.2**フォルダがインストールされます。

4D Client**6.5.2**フォルダには、4D Clientアプリケーションと関連ファイルがあります。

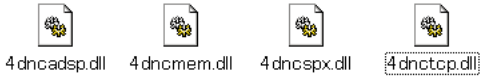


4D Clientを起動するには、“ 4DCLIENT.EXE ”アイコンをダブルクリックします。

ネットワークコンポーネント

サーバマシンとクライアントマシンの両方に、インストーラプログラムは4Dネットワークコンポーネントもインストールし、それによって4D Serverと4D Clientはネットワーク経由で通信できるようになります。ネットワークコンポーネントはアクティブなWindowsフォルダ内のACI\Networkフォルダにインストールされます。ACI\Networkフォルダには、少なくとも次のファイルがあるはずです。

■ 4Dncadsp.dllはAppleTalkのネットワークコンポーネントです。



■ 4Dncspx.dllはNovell IPX/SPXのネットワークコンポーネントです。

■ 4Dnctcp.dllはTCP/IPのネットワークコンポーネントです。

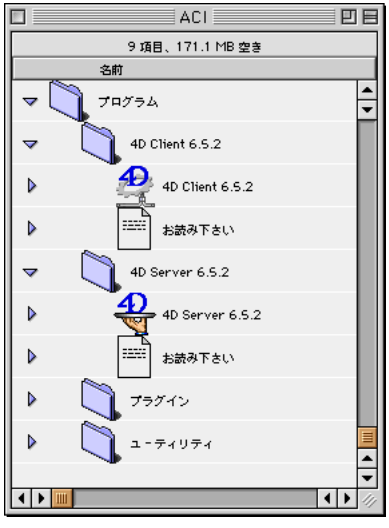
ネットワークコンポーネントの詳細については、『4D Server ネットワークコンポーネント』マニュアルの第1章「はじめに」の「ネットワークプロトコルとネットワークコンポーネント」を参照してください。

Macintoshの場合

インストールプログラムで示されたデフォルトフォルダに4D Serverと4D Clientをインストールした場合には、次に示すフォルダとファイルがディスク上に作成されています。

4D Server

4D Server 6.5.2アプリケーションは、4D Server 6.5.2フォルダに配置され、このフォルダは**ACIプロダクト:プログラム**フォルダ内にあります。



4D Client

4D Client 6.5.2アプリケーションは、4D Client 6.5.2フォルダに配置され、このフォルダは**ACIプロダクト:プログラム**フォルダ内にあります。

ネットワークコンポーネント

Macintoshでは、ネットワークコンポーネントは4D Serverアプリケーションと4D Clientアプリケーションに統合されています。

次のステップへ

マシンがネットワーク経由で通信を行うように設定されていることに注意してください。使用するネットワークプロトコルがわからない場合には、『4D Server ネットワークコンポーネント』マニュアルの第1章「はじめに」の「ネットワークプロトコルとネットワークコンポーネント」の節、「プロトコルとプラットフォームの組み合わせ」の節を参照してください。

4D Server、4D Client、およびネットワークコンポーネントが正常にインストールされたら、「サーバデータベースの作成」の節に進んでください。インストールで不具合が発生した場合で、前述のファイルのいずれかが存在しない場合には、『ACI Product Lineインストールガイド』を参照し、これらのファイルのインストールをしてから先に進んでください。

サーバデータベースの作成

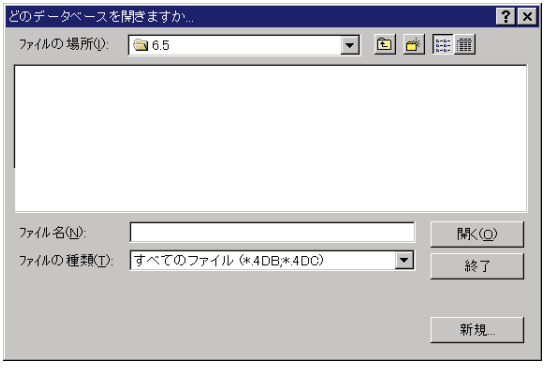
この節では、4D Clientを使用し、次にWebブラウザを使用してネットワークにアクセスできるサーバデータベースの作成方法を説明します。初めて4D Serverと4D Clientを使用する場合には、その前にインストール環境を確認することをお勧めします。確認するためには、「インストール環境のチェック」の節を参照してください。

サーバデータベースを作成、または開くには、4D Serverを起動します。

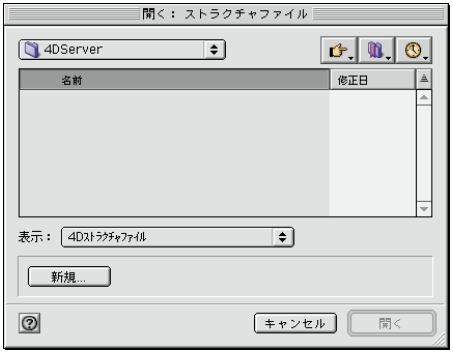
1. 4D Serverアイコンをダブルクリックして4D Serverを起動する。

Windowsの場合は、“ 4DSERVER.EXE ”アイコンをダブルクリックします。
Macintoshの場合は、“ 4D Server 6.5.2 ”アイコンをダブルクリックします。

「データベースを開く...」ダイアログボックスが表示され、既存のデータベースを開くか、新規データベースを作成するかを選択することができます。ここでは、新しくデータベースを作成することにします。



Windows版の「データベースを開く」ダイアログボックス



Macintosh版の「データベースを開く」ダイアログボックス

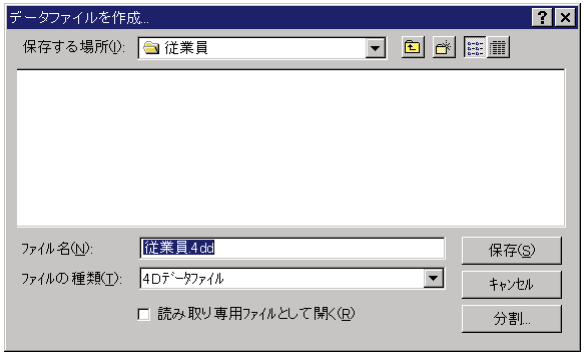
2. **「新規」ボタンをクリックして新しいデータベースを作成する。**
「データベース名」ダイアログボックスが表示され、新しいデータベースの名前を入力できます。

Windows版の「データベース名」ダイアログボックス

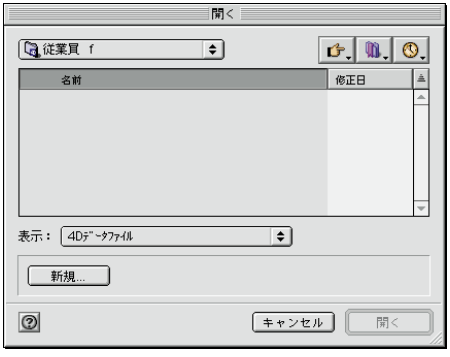
Macintosh版の「データベース名」ダイアログボックス

3. **データベースのストラクチャファイルに名前をつける。**
“ 従業員 ” と入力してから「保存」をクリックします。

4. **データベースのデータファイルに名前をつける。**
「データファイル作成」ダイアログボックスが表示され、データファイルを作成するように要求されます。4D Serverによりデフォルトの名前がデータファイルに設定されます。



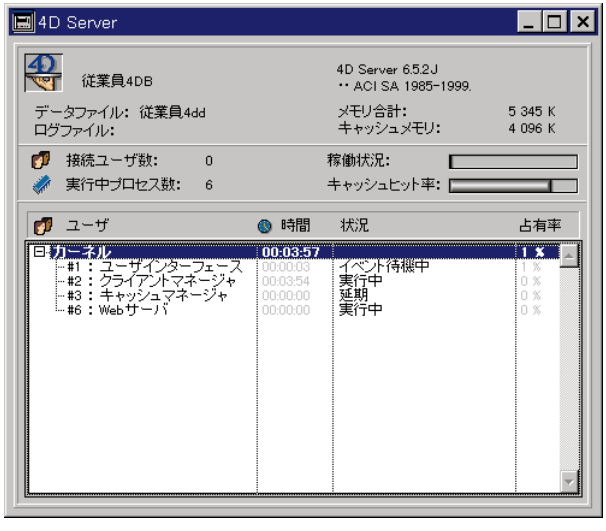
Windows版の「データファイル作成」ダイアログボックス



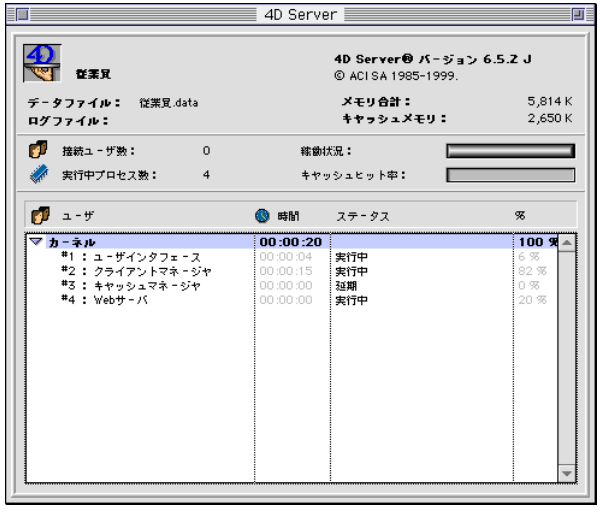
Macintosh版の「データファイル作成」ダイアログボックス

「保存」をクリックします。

4D Serverはデータベースを作成し、「プロセス」ウインドウを表示します。



Windows版の「4D Server プロセス」ウインドウ



Macintosh版の「4D Server プロセス」ウインドウ

「4D Server プロセス」ウインドウは2つの部分に分れます。

上側の部分にはサーバについての一般的な情報が表示され、下側の部分には現在のサーバの使用状況についての情報が表示されます。

この時点の接続ユーザ数はゼロになっています。これは、まだクライアントがデータベースに接続していないことを示しています。現在実行中のプロセスの数は4つです。これらの4つのプロセスは、データベースエンジン（カーネル）およびビルトインWebサーバによって作成されたプロセスです。

4D Serverウインドウの下側の部分は、接続ユーザと現在実行中のプロセスを示します。クライアントワークステーションが接続されていない場合には、サーバエンジンの使用状況だけが表示されます。

次のステップへ

この時点で、データベースは、TCP/IP、Novell社のIPX/SPX、AppleTalkのいずれかのプロトコルを使用し、ネットワーク経由でWindowsまたはMacintoshの4D Clientから接続することができます。しかし、データベースはWeb接続の準備はまだできていません。これは、少なくとも1つのメニューバーまたは「On Web Connection」データベースメソッドが必要となるためです。

「4D Clientからサーバデータベースへ接続する」の節に進んでください。このチュートリアルでは、まず最初に4D Clientを使用して接続し、データベースのストラクチャを定義してから、データベースにレコードを追加します。

4D Clientからサーバデータベースへ接続する

初めて4D Serverと4D Clientを使用する場合には、その前にインストール環境を確認することをお勧めします。確認するためには、「インストール環境のチェック」を参照してください。

この節では、次の事項について説明します。

- 作成したサーバデータベースにクライアントを接続する方法
- データベースストラクチャの作成。この節には、データベースへのテーブルやフィールドの作成、新規レコードの入力、既存レコードの変更についてのチュートリアルも含まれています。
- セカンドユーザの接続
- 同時使用

データベースへの接続

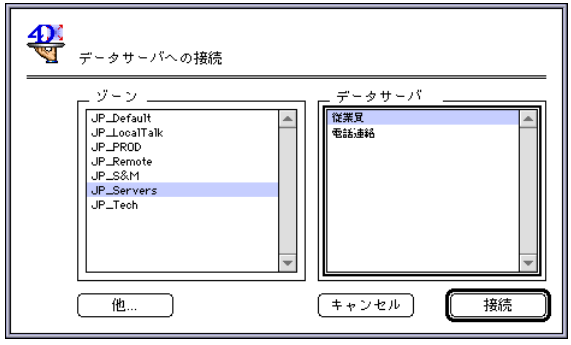
データベースは4D Server（「サーバデータベースの作成」の節参照）で作成しましたが、データベースの設計と実際のデータに対する変更は、すべてクライアントマシンから行います。この節では、サーバに接続してサーバデータベースを開く方法について学習します。

1. 4D Clientアプリケーションアイコンをダブルクリックする。

Windowsの場合は、“ 4DCLIENT.EXE ”アイコンをダブルクリックします。
Macintoshの場合は、“ 4D Client 6.5.2 ”アイコンをダブルクリックします。

マシン（WindowsまたはMacintosh）、インストールしたネットワークコンポーネント、ネットワーク設定に応じて、「接続」ダイアログボックスのいずれか1つが表示されます。

Windows版の「TCP/IP 接続」ダイアログボックス



Macintosh版の「AppleTalk 接続」ダイアログボックス

2. **データベースの名前である“ 従業員 ”を選択し、「OK」をクリックする。**

選択したデータベースがクライアントワークステーション上で開かれます。データベースは「デザイン」環境で開かれるので、データベースのストラクチャを作成することができます。

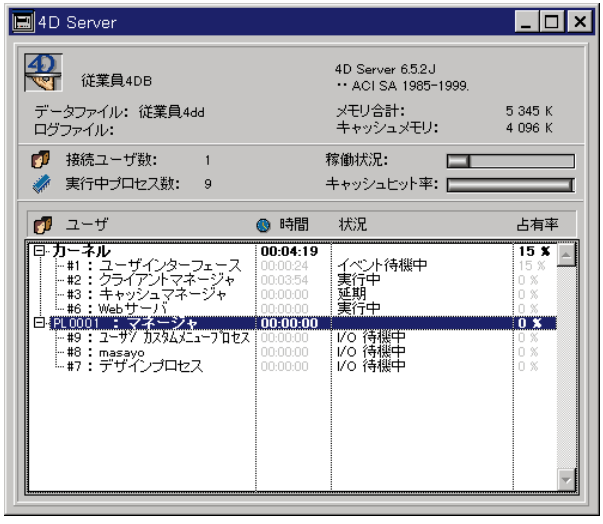
Tips：4D Server上で、作成したデータベース名が表示されない場合には、次の点を確認してください。

- もう一方のマシンで4D Serverはまだ実行中ですか?
- 使用しているマシンはネットワークに接続されていますか?
- 「接続」ダイアログボックスの使い方がよくわからない場合には、「4D Serverデータベースへの接続」の節を参照してください。
- 操作しているマシンのネットワークプロトコル設定がよくわからない場合には、『4D Serverのネットワークコンポーネント』マニュアルの第1章「はじめに」の「ネットワークプロトコルとネットワークコンポーネント」の節、「プロトコルとプラットフォームの組み合わせ」の節を参照してください。また、使用しているプロトコルに対応するこのマニュアルの節も参照してください。

サーバの使用状況

「4D Server プロセス」ウインドウを見ると、自分のネットワークユーザ名がユーザリストに追加され、接続ユーザ数が1になっています。

現在、6つのプロセスが実行されています。



最初の4つは4D Server自体のプロセスで、4D Serverの起動時に生成されます。2つの新しいプロセスが、最初にサーバに接続したユーザのプロセスです。

- デザインプロセスは、「デザイン」環境を管理します。「ファイル」メニューから「閉じる⇒ストラクチャ」を選択してクライアントマシン上のデザインプロセスを閉じると、サーバ上のデザインプロセスは終了します。
- ユーザ／ランタイムプロセスは、「ユーザ」モードと「ランタイム」モードを管理します。ユーザが増えるたびに、少なくとも2つのプロセスがリストに追加されます。ユーザ名の左側にある＋印（Windows）または三角印（Macintosh）をクリックして、ユーザのプロセスを隠すことができます。プロセスを再び表示させるには、＋印（Windows）または三角印（Macintosh）を再度クリックします。

次のステップへ

接続が完了したので、シングルユーザ環境で4th Dimensionと同じ機能を使い、データベース作業が行えます。まず最初に、ストラクチャを定義する必要があります。「データベースストラクチャの定義」の節に進んでください。

データベースストラクチャの定義

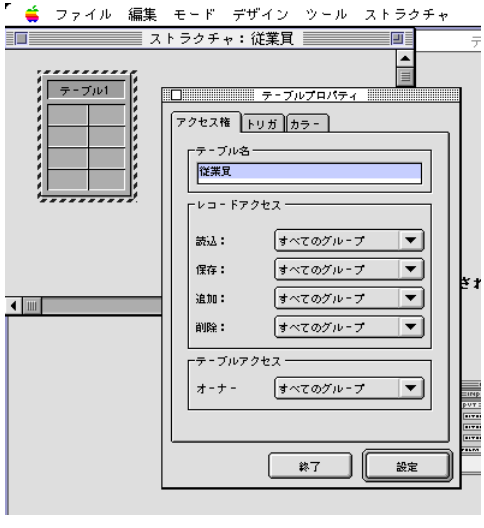
サーバデータベースに接続した後（「4D Client からサーバデータベースへ接続する」の節参照)、「ストラクチャ」ウインドウを前面に表示させます。

新しいデータベースには、最初のテーブル名がデフォルトで[テーブル1]と設定されています。

[従業員]テーブル定義の設定－例

1. **[テーブル1]のタイトルバーをダブルクリックする。**
   または
   [テーブル1]のタイトルバーをクリックし、「ストラクチャ」メニューから「テーブルプロパティ」を選択する。
   「テーブルプロパティ」フローティングウインドウが表示されます。

2. **「テーブル名」領域で “ 従業員 ” と入力してから、「設定」ボタンをクリックする。**



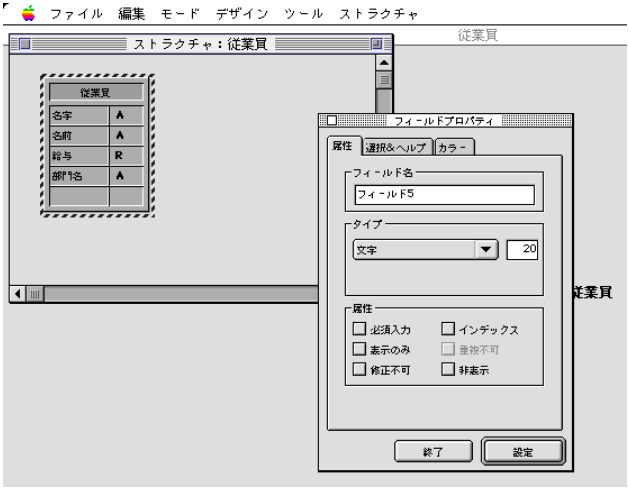
   テーブルの名前を変更した後で、「テーブルプロパティ」フローティングウインドウを閉じます。

3. **[従業員]テーブル上の現在は空のフィールドリストをダブルクリックする。**
   または
   [従業員]テーブル上の現在は空のフィールドリストをクリックし、「ストラクチャ」メニューから「新規フィールド」を選択する。
   「フィールドプロパティ」フローティングウインドウが表示されます。

4. **[従業員]テーブルに次のフィールドを追加する。**

フィールド名	フィールドタイプ
名字	文字（20桁）
名前	文字（20桁）
給与	実数
部門名	文字（20桁）

各フィールドに対して、「フィールド名」エリアにフィールドの名前を入力してから、フィールドタイプを選択し、「設定」をクリックします。



フィールドをテーブルに追加した後で、「フィールドプロパティ」ウインドウを閉じます。

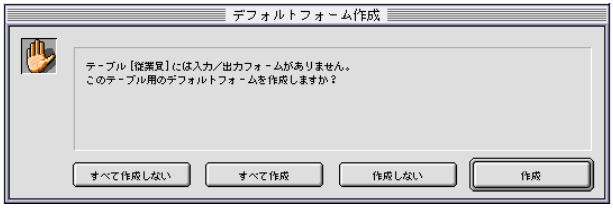
注：他の4D Clientがこのサーバデータベースに対して同時に作業を行っている場合は、作成したフィールドが数秒で他のクライアントマシン上に表示されます。変更はリアルタイムでサーバ上に反映されますが、頻繁に画面が更新されるのを避けるために、すぐに他の画面には現れません。

[従業員]テーブル用フォームの作成

[従業員]テーブルを定義した後には、レコードを追加し、作業するフォームが必要になります。この場合、新規フォームウィザードを使用すると、フォームを自由に作成できます。しかし、4D Serverにはデフォルトの入力フォームと出力フォームを即座に作成できる便利なショートカットが用意されています。

1. **「モード」メニューから「ユーザ」を選択する。**
「ユーザ」モードに切り替わります。4D Serverは、テーブルにフォームが存在しないことを検知し、ユーザに対してフォームを作成するかどうかを質問します。

2. **「作成」をクリックする。**



これで、レコードを1つずつ追加する、または表示する入力フォームと、一覧モードで複数のレコードを表示する、または入力するための出力フォームが作成されます。

次のステップへ

これで、サーバデータベースでデータを操作する準備ができました。「4D Serverでのデータ操作」の節に進んでください。

4D Serverでのデータ操作

「データベースストラクチャの定義」の節では、[従業員]テーブルを作成し、4D Serverにより、このテーブル用のデフォルトフォームが作成されました。これでレコードを入力する準備ができました。

レコードの入力

「ユーザ」モードでは、レコードの入力、検索、プリント、変更を行います。まだ「ユーザ」モードに切り替えていない場合には、「モード」メニューから「ユーザ」を選択します。4D Serverから「レコードが登録されていません：従業員.」というメッセージが表示されます。レコードがある場合には、4D Serverにより自動的に作成されたデフォルト出力フォームを使い、画面上に表示されます。

1. **「更新」メニューから「新規レコード」を選択する。**
   空白の入力フォームが表示されます。

2. **最初のレコードを入力する。**
   「タブ」キーかマウスを使用して、フィールド間を移動します。

3. **ディスクアイコン（二番目のアイコン）をクリックし、データ入力を確定する。**
   空白の入力フォームが表示されるので、続けて新しいレコードを追加します。

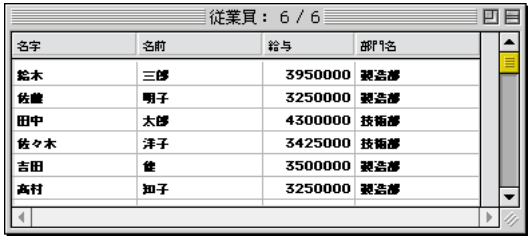
4. **さらに5つのレコードに次の値を入力します。**

名字	名前	給与	部門名
佐藤	明子	3250000	製造部
田中	太郎	4300000	技術部
佐々木	洋子	3425000	技術部
吉田	健	3500000	製造部
高村	知子	3250000	製造部

5. **最後のレコードを入力したら、キャンセルアイコン（ディスクアイコンの上にあるアイコン）をクリックして、新しい空白の入力フォームをキャンセルする。**
制御は出力フォームに戻ります。

6. **6レコードすべてが表示されない場合には、「クエリ」メニューから「すべてを表示」を選択する。**

次の画面が表示されます。



これで、レコードはサーバマシン上のデータベースに保存されました。2番目の4D Clientがサーバマシンに接続すると、今追加したレコードが表示されます。逆に、他のクライアントもレコードを入力していた場合には、「クエリ」メニューから「すべてを表示」を選択すると、他のクライアントが入力したものも含めて、すべてのレコードを表示できます。サーバに保存されたレコードは、すべてのユーザからアクセス可能です。

レコードの検索

[従業員]テーブルにレコードを入力したら、レコードの検索、ソート、プリント、または操作ができます。例えば、“ 技術部 ” 部門の従業員を検索してみましょう。

1. **「クエリ」メニューから「検索...」を選択する。**
   クエリエディタが表示されます。

2. **「フィールド」リストに表示されている “ 部門名 ” をクリックする（上図参照）。**
   部門名がウインドウの上側のクエリ定義エリアに表示されます。

3. **「比較演算子」リストから “ = ” をクリックする（上図参照）。**
   ウインドウの上の部分に “ = ” が表示されます。

4. **「値」エリアに “ 技術部 ” と入力する。**

5. **「クエリ」をクリックする。**

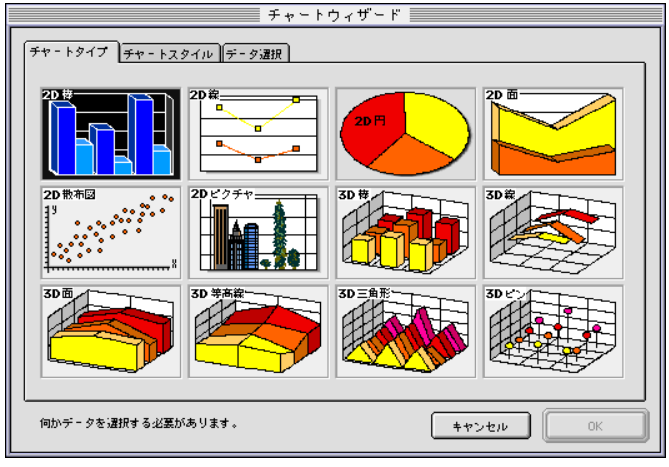
   クエリが4D Serverに送られると、4D Serverは4D Clientに応答を返します。出力フォームには、“ 技術部 ” 部門で働いている従業員だけが表示されます。

6. **全レコードをもう一度表示するには、「クエリ」メニューから「すべてを表示」を選択する。**

チャートの作成

全レコードが表示されていない場合には、「クエリ」メニューから「すべてを表示」を選択します。

1. **「レポート」メニューから「チャート...」を選択する。**
   チャートウィザードが表示されます。



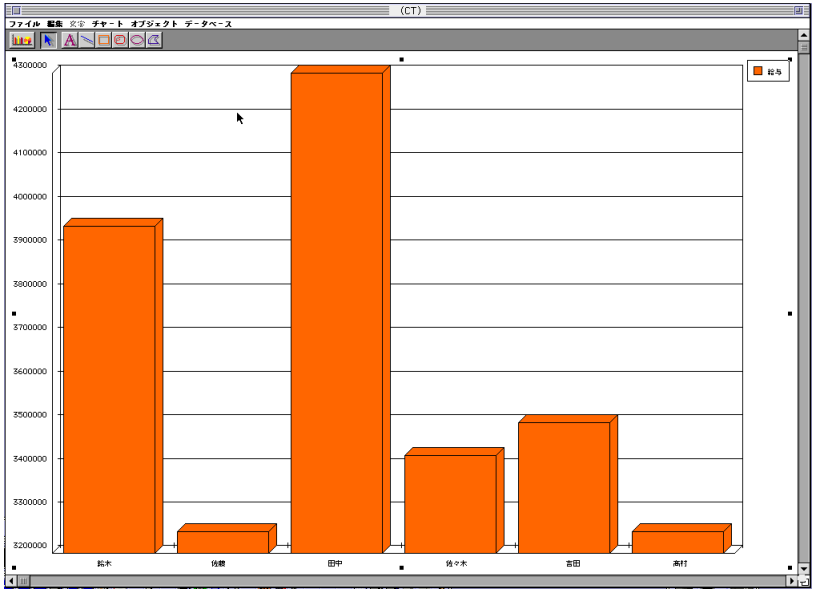
   これから2D棒チャート（デフォルトのチャートタイプ）を作成します。

2. **「データ選択」タブをクリックする。**
   チャートウィザードの3ページ目が表示されます。このページで、新しいチャートを作成するためのデータを選択できます。

3. **図のように、「項目軸（X）」エリアに「名字」フィールドをドラッグ&ドロップする。**

4. **図のように、「数値軸（Z)」リストに「給与」フィールドをドラッグ＆ドロップする。**

5. **「OK」をクリックする。**
「チャート」ウインドウが現れ、指定した設定に応じて作成されたチャートが表示されます。



次のステップへ

数分で、サーバデータベースの作成、テーブルの定義、レコードの追加を行い、その後でデータベースに入力したデータを使用して検索とチャートの作成を行いました。

次に、データベースにカスタムメニューバーを追加してみましょう。「カスタムメニューバーの追加」の節に進んでください。

カスタムメニューバーの追加

この節では、2つのメソッドとカスタムメニューバーを作成します。つまり、4Dのカスタムアプリケーションを作成します。

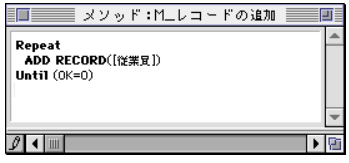
2つのメソッドの追加

1. **「モード」メニューから「デザイン」を選択して「デザイン」モードに戻る。**

2. **「デザイン」メニューから「新規メソッド」を選択する。**
「新規メソッド」ダイアログボックスが表示されます。

3. **「新規メソッド」ダイアログボックスに“ M_レコードの追加 ”と入力してから「OK」をクリックする。**
「メソッド：M_レコードの追加」というタイトルで、空のメソッドエディタウインドウが表示されます。

4. **次のように“ M_レコードの追加 ”メソッドのコードを入力する。**



5. **「デザイン」メニューから「新規メソッド」を選択する。**
「新規メソッド」ダイアログボックスが表示されます。

6. **「新規メソッド」ダイアログボックスに“ M_レコードの一覧 ”と入力してから「OK」をクリックする。**
「メソッド：M_レコードの一覧」というタイトルで、空のメソッドエディタウインドウが表示されます。

7. **次のように“ M_レコードの一覧 ”メソッドのコードを入力する。**

これで2つのメソッドが作成できました。これからカスタムメニューバーを作成し、カスタムメニューコマンドにこの 2 つのメソッドを付加します。

カスタムメニューバーの追加

1. **「デザイン」モードの「ツール」メニューから「メニューバーエディタ」を選択する。**
   メニューバーエディタが表示されます。

2. **左下端にある「追加」ボタンをクリックして、“ メニューバー #1 ” を作成する。**
   “ メニューバー #1 ” が左側のリストに現れます。右側のリストには「ファイル」メニューが自動的に表示されます。

3. **“ メニューバー #1 ” に新しいメニューを追加するために「メニュー追加」ボタンをクリックする。**

4. **メニュータイトルとして “ チュートリアル ” と入力してから「Enter」キーを押す。**

5. **「項目追加」をクリックして、「チュートリアル」メニューにメニューコマンドを追加する。**

6. **メニューコマンドテキストとして “ 従業員の追加... ” と入力し、「Enter」キーを押す。**

7. **「項目追加」をクリックして、「チュートリアル」メニューに2番目のメニューコマンドを追加する。**

8. **メニューコマンドテキストとして “ 従業員の検索... ” と入力し、「Enter」キーを押す。**

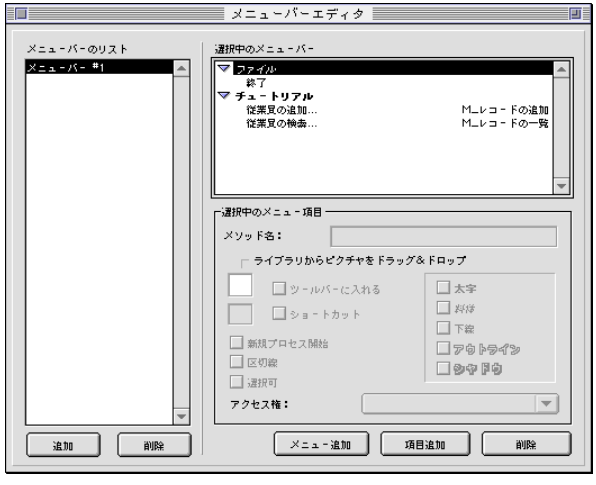
“ メニューバー #1 ” は次のように表示されるはずです。

9. **「従業員の追加...」メニューコマンドをクリックして、「メソッド名」エリアに “ M_レコードの追加 ” と入力する。**

10.「従業員の検索...」メニューコマンドをクリックして、「メソッド名」エリアに“ M_レコードの一覧 ”と入力する。

“ メニューバー #1 ”は、次のようになるはずです。

「メニューバーエディタ」ウインドウ全体は、次のようになります。



11.「メニューバーエディタ」ウインドウを閉じる。

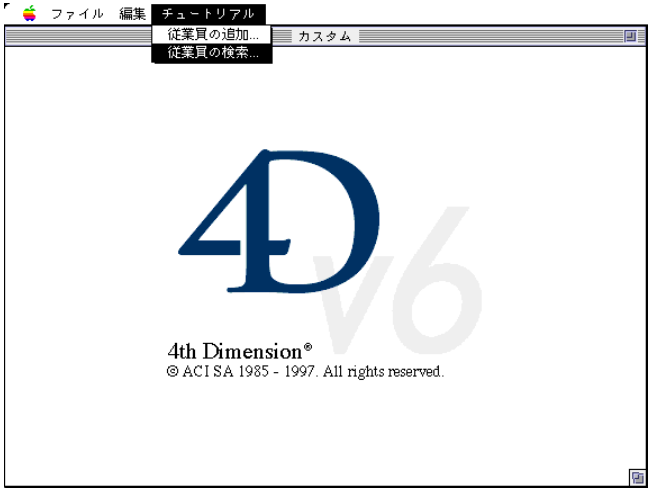
これで完成です。

12.「モード」メニューから「ユーザ」を選択して、「ユーザ」モードに戻る。

「モード」メニューには3番目のメニューコマンド「カスタム」が追加されていることに注意してください。

13.「モード」メニューから「カスタム」を選択する。

作成したメニューを使い、アプリケーションを使用してみましょう。



例えば、「チュートリアル」メニューから「従業員の検索...」を選択すると、クエリエディタ（「ユーザ」モードの組み込みクエリエディタ）が表示されます。ここで検索条件を定義し、検索結果レコードを表示する、または変更することができます。

興味深い点は、知らないうちに2つのアプリケーションを開発していることです。

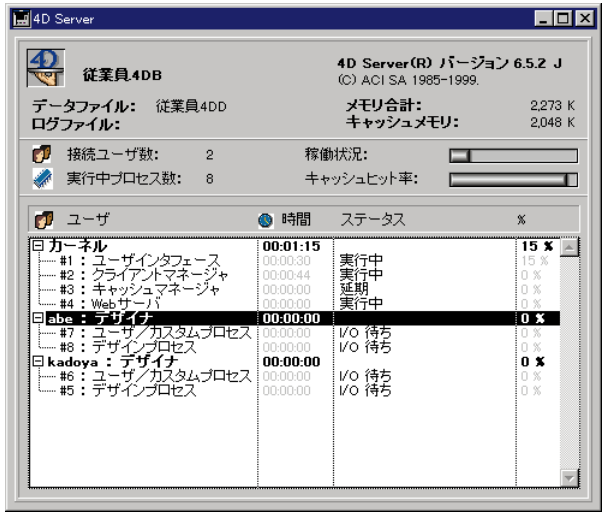
この理由を確認するために、「4D Serverで同時作業を行う」の節に進んでください。

4D Serverで同時作業を行う

Windows上でこのチュートリアルを実行している場合、Macintosh上でこのサーバデータベースをそのまま使用することができます。Macintosh上でこのチュートリアルを実行している場合（ここで行っているように）、このサーバデータベースをそのままWindows上でも使用することができます。

セカンドユーザがサーバデータベースに接続する

このチュートリアルでは、Windowsの4D Clientからサーバデータベースに接続します。接続するとすぐに、4D Serverの「プロセス」ウインドウにセカンドユーザが接続したことが示されます。



Windows上のクライアント側では、もう一方のプラットフォーム上での作業結果はすべて即座に利用できます。

次に「デザイン」モードを示します。

これが、今までに作成した6つのレコードと2つのメソッドです。次に、「カスタム」モードに移動します。「チュートリアル」メニューが使用可能になっています。

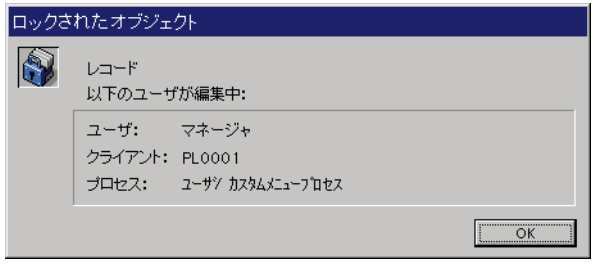
レコードの同時作業

1. **2番目のクライアントマシンで、「チュートリアル」メニューから「従業員の検索...」を選択して、“ 部門名が “ 技術部 ” である ” レコードを検索する。**

2. **1番目のクライアントマシンでも同じことを行う。**
   どちらのマシンでも、2つのレコードで構成されるリストが表示されます。

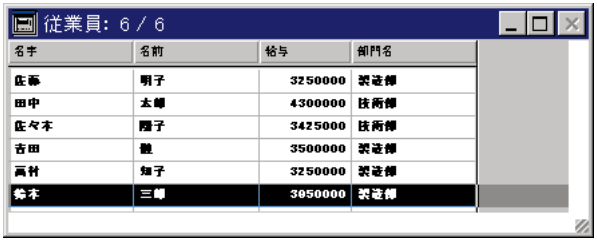
3. **2番目のマシンで、“ 佐々木、洋子 ” というレコードをダブルクリックする。**
   画面は次のようになります。

4. **同じことを1番目のマシンでも行う。**
   4D Serverにはレコードロック機能が組み込まれているので、そのレコードが既に使用中であるという警告が表示されます。

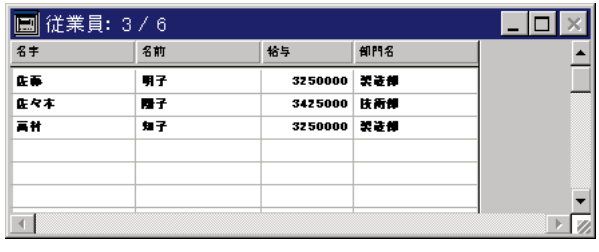


   しかし、リードオンリーモード（表示はできるが、変更はできない）で、このレコードにアクセスすることができます。

5. **2番目のマシンで、名前を“陽子”に変更し、その変更を確定する。**
   リストは新しい名前で更新されています。



6. **1番目のマシンで、入力フォームでのレコード表示をキャンセルする。**
   1番目のマシンのリストも新しい名前で変更されています。



デザインオブジェクトの同時作業

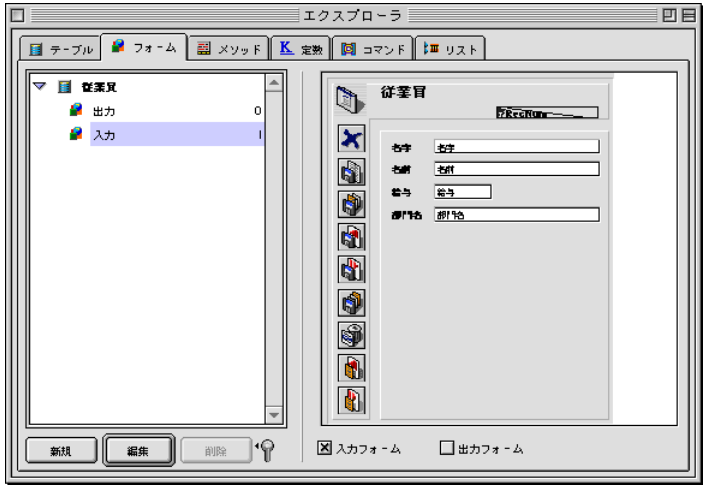
4D Serverはデータサーバであると同時に、アプリケーションサーバでもあります。この意味を実際に検討してみましょう。

1. **1番目のマシンで、「終了」をクリックしてレコードリストを閉じる。「ファイル」メニューから「終了」を選択して、「ユーザ」モードに戻る。次に「モード」メニューから「デザイン」を選択して「デザイン」モードに戻る。**

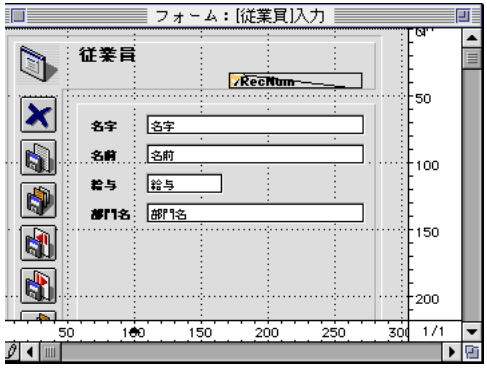
2. **2番目のマシンでも同じことを行う。**

3. **1番目のマシンで、「ツール」メニューから「エクスプローラ」を選択する。**

   「エクスプローラ」ウインドウが現われます。



4. **「フォーム」タブをクリックする。「従業員」テーブルを展開する。「入力」フォームを選択して「編集」をクリックする。**
「入力」フォーム用に「フォームエディタ」ウインドウが表示されます。

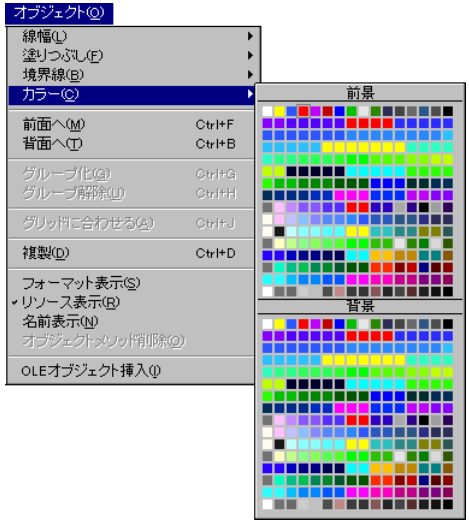


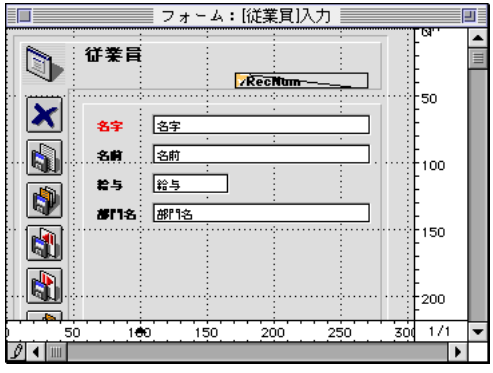
5. **2 番目のマシンでも同じことを行う。**
フォームは1番目のマシンで既に「編集」モードになっているので、4D Serverの組み込みオブジェクトロック機能により、次のメッセージが表示されます。

しかし、2番目のマシンは表示モードなのでフォームを編集することはできません。オブジェクトを選択し、それを別のフォームにコピーすることはできますが、フォーム自体を変更することはできません。フォームの左下端のペンアイコンに×印がついていることに注意してください。このアイコンは、フォームがを変更できないということを示しています。

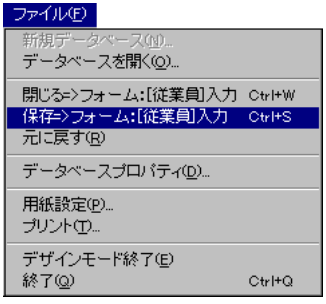
6. **1番目のマシンで、“ 名字 ” フィールドの左側にある “ 名字 ” ラベルを選択する。**

7. 「オブジェクト／カラー」階層メニューを使用して、次のようにこのオブジェクトの色を赤に設定する。





8. 「ファイル」メニューから「保存...」を選択する。



9. 2番目のマシンで、再ロードするためにフォームを閉じた後、再度開く。

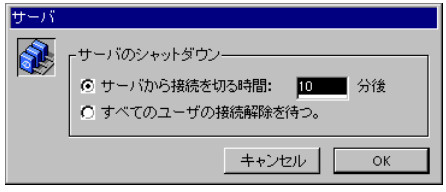
これで、1番目のマシンで行われた変更が、このマシンでも利用可能になりました。

4D Serverでは、他のユーザと同時にデータベースの開発を行えます。

サーバのシャットダウン

4D Serverは、同一レコードやオブジェクトに対して同時にアクセスをしている4D Clientユーザに通知を行うだけではなく、終了時にネットワーク経由で警告メッセージを送る機能を備えています。

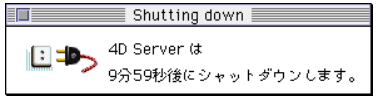
1. **2つのクライアントをサーバデータベースに接続したままで、サーバマシン上で「ファイル」メニューから「終了」を選択する。**
   「サーバのシャットダウン」ダイアログボックスが表示されます。



2. **「OK」をクリックする。**
   ほぼ同時に、2つのクライアントマシンはサーバがシャットダウンするという通知を受け取ります。例えば、あるクライアントがレコードを追加している場合、データ入力を終了し、確定する時間は十分にあります。

   注：他の方法として、“ すべてのユーザの接続解除を待つ ” オプションを使用してサーバをシャットダウンすることもできます。

Windowsクライアントマシン上のシャットダウン警告メッセージ



Macintoshクライアントマシン上のシャットダウン警告メッセージ

3. **サーバのシャットダウンが行われている間に、2 つのクライアントマシンで4D Clientを終了する。**

次のステップへ

ここで、これまでの9分間の作業が終り、サーバのシャットダウンが行われている間に、休憩したい方もいるでしょう。ここで注目する点は、2つのアプリケーションを開発したばかりでなく、実は3つのアプリケーションを開発したのだということです。

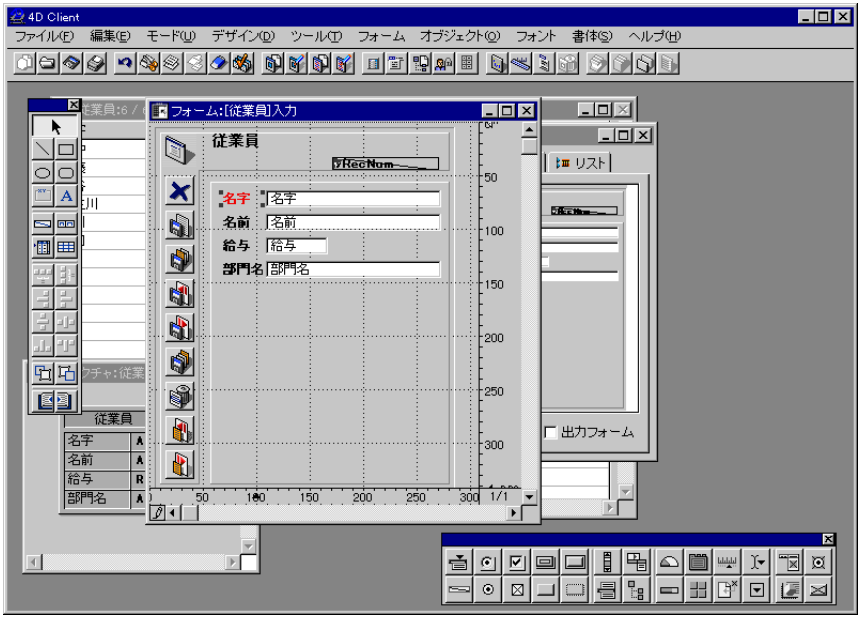
この理由を知るには、「4D ServerはWebサーバである」の節に進んでください。この節を終了するには約1分間かかります。

4D ServerはWebサーバである

1. 再度4D Serverを起動して、サーバデータベースを再起動する。

2. 作成した“従業員”データベースを開く。

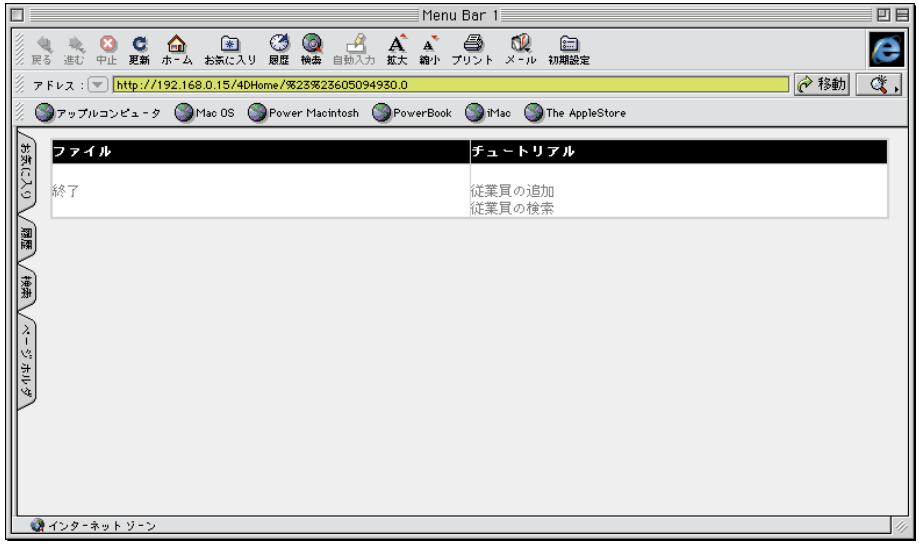
3. 4D Clientを使用してデータベースに接続する。



サーバマシン上には、接続中のユーザが1人います。

4. **別のマシンから、Webブラウザプログラム（例：Netscape社のNetscape NavigatorTM またはマイクロソフト社のInternet Explorer）を起動する。**

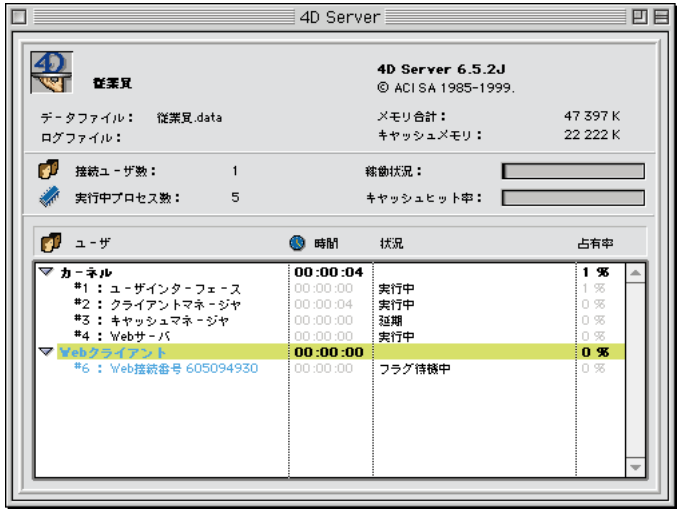
5. **ブラウザのURL入力エリアにサーバデータベースのIPアドレスを入力してから、「Enter」キーを入力する。**
   次のようにWeb版のカスタムメニューバーが表示されます。



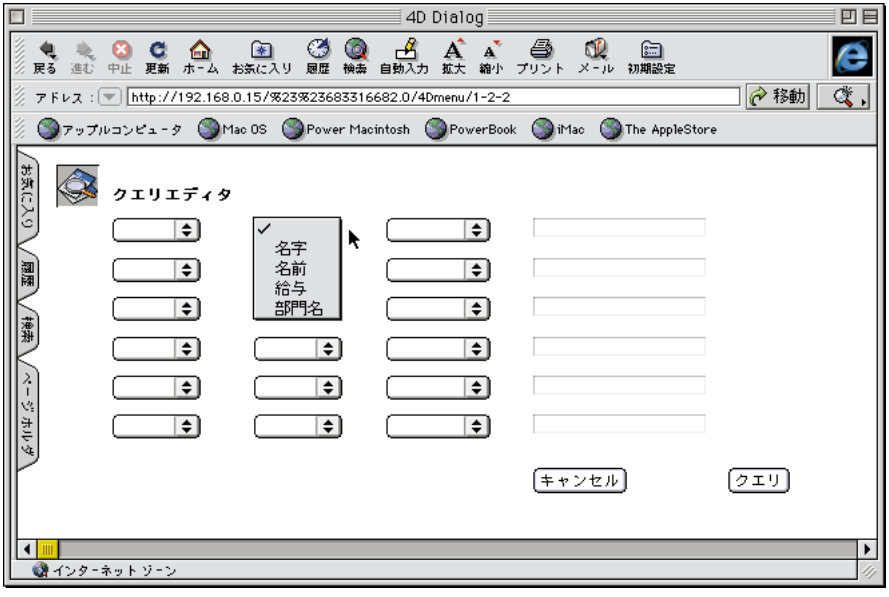
この結果を得るために、HTMLコードを書いたり、CGIモジュールを作成する必要がないという点に注意してください。つまり、4D ServerはWebサーバでもあるのです。

サーバマシン上では、Web接続が「プロセス」ウインドウに表示されます。

6. **Webブラウザ側で、“ 従業員の検索 ” をクリックする。**



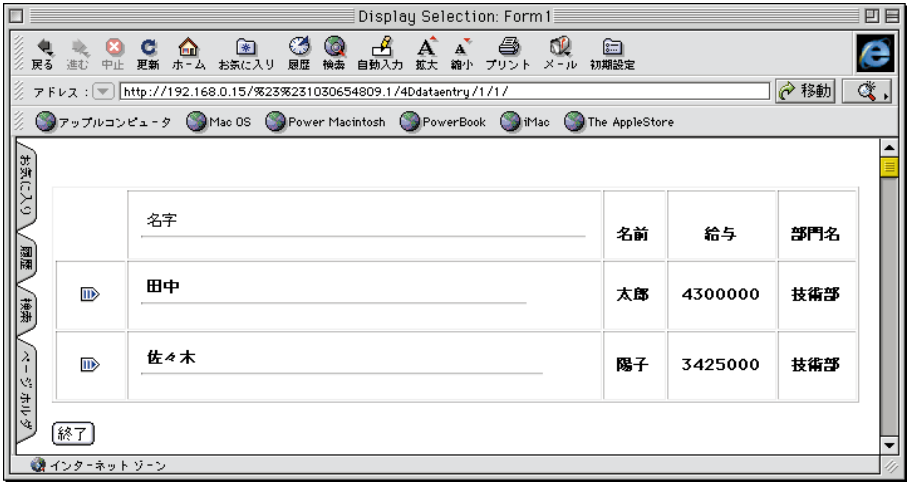
このWebページは次のようになります。



4D Serverは、実行中に標準の「クエリ」エディタウインドウをHTMLページに変換しました。

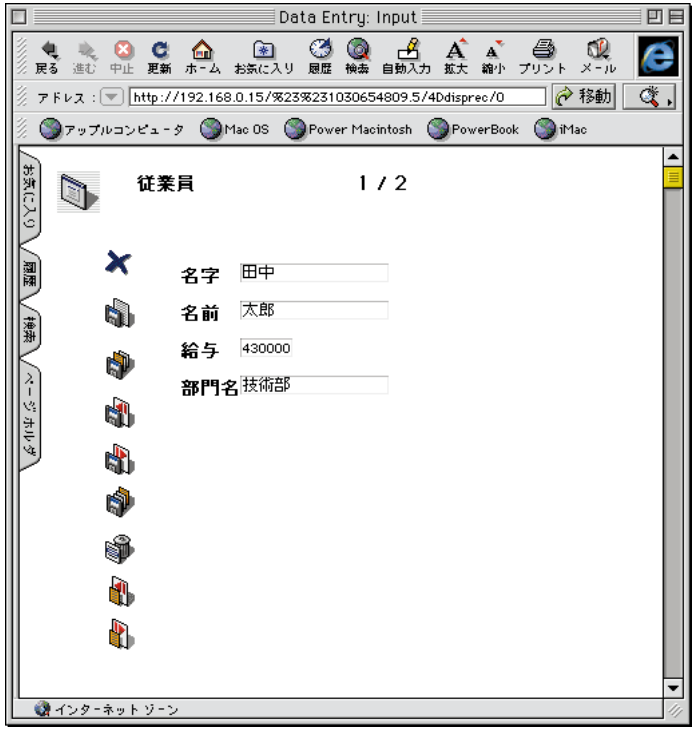
7. 検索条件を“部門名＝“技術部””と設定する。

検索結果は次の通りです。



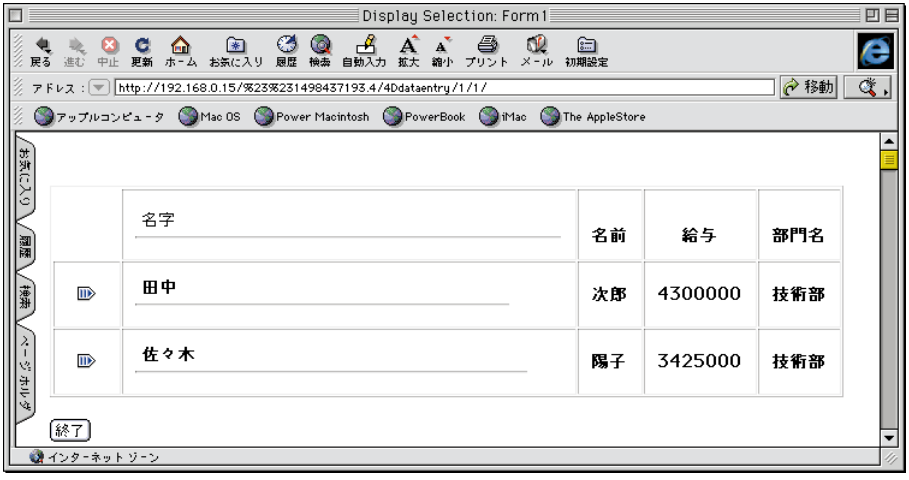
4D Serverは、4Dデータ をWebデータにダイナミックに変換し、レコード一覧を表示します。

8. **左側のカラムのアイコンを使用して、“ 田中、太郎 ” レコードを開く。**
4D Clientで使用していた入力フォームがWebページ（4D Serverにより変換された）として表示されます。



9. **名前を “ 次郎 ” に変更し、そのデータ入力を確定する。**

10. レコード一覧に戻り、変更がレコード一覧に反映されていることを確認する。



11. 最後に、4D Clientマシンの「ユーザ」モードに移る。

これまで行った変更がクライアントからも利用できることがわかります。

まとめ

このチュートリアルを通して（休憩を取るなど、10分以上かかることもあるでしょうが)、4D Serverを使用するのがどんなに簡単かがお分かりになったことだと思います。

- データベースを最初から作りました。
- テーブルを定義し、4D Serverにフォームを作成させました。
- レコードを数件追加し、操作しました。
- 独自のメニューバーでアプリケーションをカスタマイズしました。
- WindowsでもMacintoshでも同時にサーバデータベースを使用しました。
- サーバをシャットダウンし、再起動しました。
- 4D ClientとWebで同時にサーバデータベースを使用しました。

実際には、1つのアプリケーションを開発しながらも、結果として3つのカスタムアプリケーション（Windows、Macintosh、およびWeb）を作成しました。さらに、シングルユーザモードでデータベースを使用する必要がある場合には、そのまま4th Dimensionや4D Runtimeで直接開くことができます。

4D Serverの詳細については、このマニュアルの「はじめに」の節を参照するか、または、4D Serverを詳細に説明している他の節を参照してください。
4D環境の詳細説明については、次を参照してください。

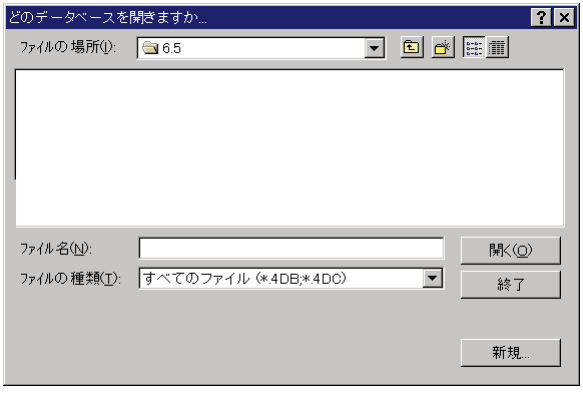
- 4Dデータベースの設計については『4th Dimensionデザインリファレンス』マニュアルを参照してください。
- クエリエディタやチャートウィザード等、このチュートリアルで使用された組み込みエディタについては『4th Dimensionユーザリファレンス』マニュアルを参照してください。
- 4D言語のコマンドについては『4th Dimensionランゲージリファレンス』マニュアルを参照してください。例えば、4D ServerのWeb機能に関しては、『4th Dimension ランゲージリファレンス』マニュアルの第61章「Web サーバコマンド」を参照してください。

新しい4D Serverデータベースの作成方法

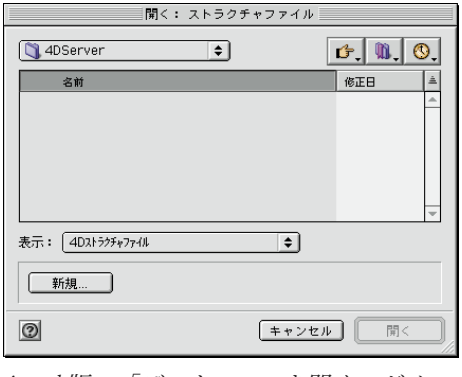
サーバデータベースを作成するには、4D Serverを起動します。

1. **4D Serverアイコンをダブルクリックして4D Serverを起動する。**
   Windowsの場合は、4DSERVER.EXEアイコンをダブルクリックします。
   Macintoshの場合は、4D Server 6.5.2アイコンをダブルクリックします。

   「データベースを開く...」ダイアログボックスが表示されるので、既存のデータベースを開くか、新規データベースを作成するかの選択ができます。

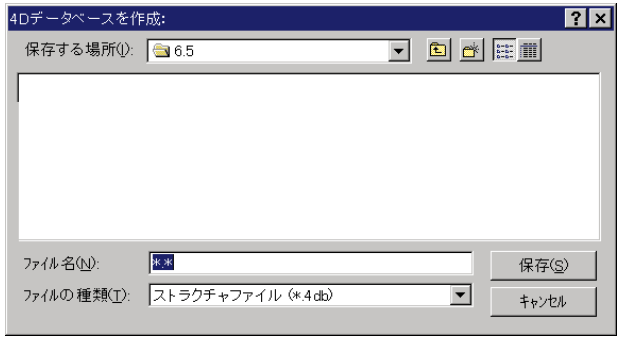


Windows版の「データベースを開く」ダイアログボックス



Macintosh版の「データベースを開く」ダイアログボックス

2. **「新規」ボタンをクリックして新しいデータベースを作成する。**
「データベース名」ダイアログボックスが表示されるので、新しいデータベースの名前を入力します。



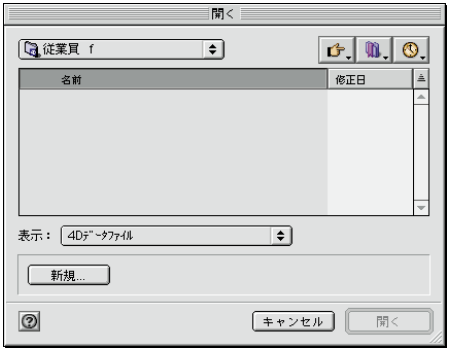
Windows版の「データベース名」ダイアログボックス

Macintosh版の「データベース名」ダイアログボックス

3. **データベースのストラクチャファイルに名前をつける。**
名前を入力してから、「保存」をクリックします。

4. **データベースのデータファイルに名前をつける。**
「データファイル作成」ダイアログボックスが表示され、データファイルを作成するように要求されます。4D Serverより、デフォルトの名前がデータファイルに設定されます。

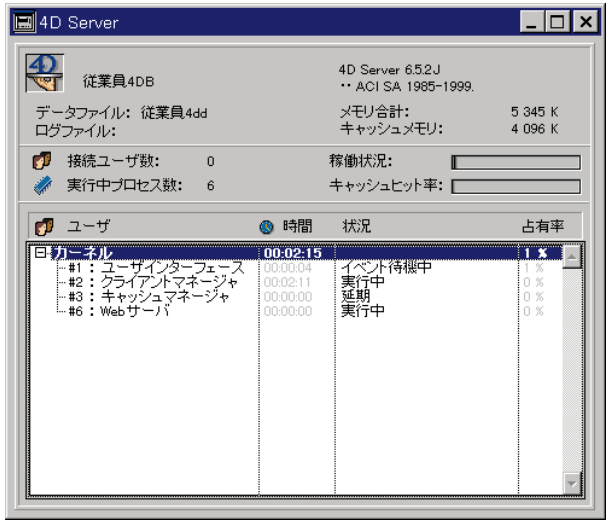
Windows版の「データファイル作成」ダイアログボックス



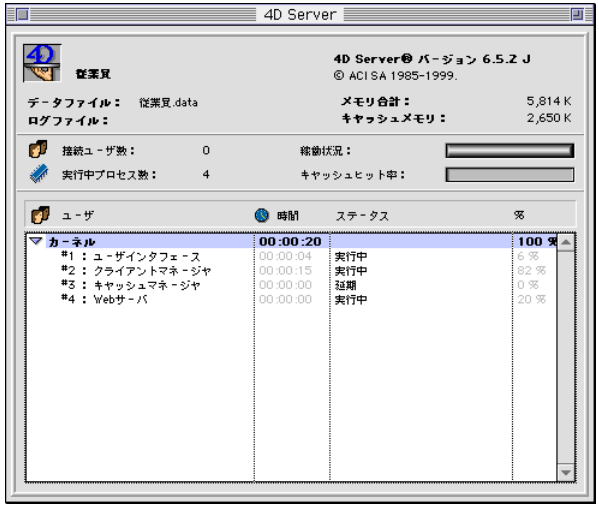
Macintosh版の「データファイル作成」ダイアログボックス

「保存」をクリックします。

4D Serverはデータベースを作成し、「プロセス」ウインドウを表示します。



Windows版の「4D Server プロセス」ウインドウ



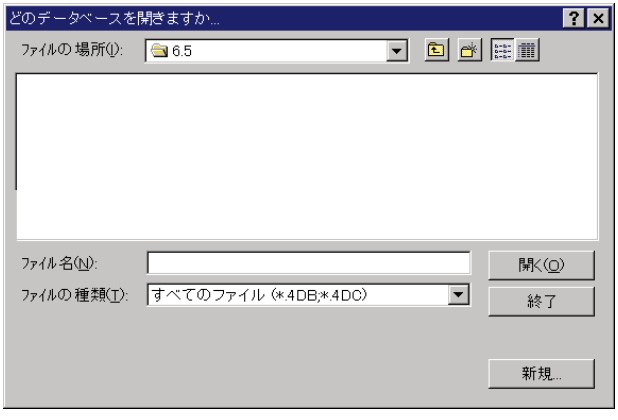
Macintosh版の「4D Server プロセス」ウインドウ

既存の4D Serverデータベースを開く

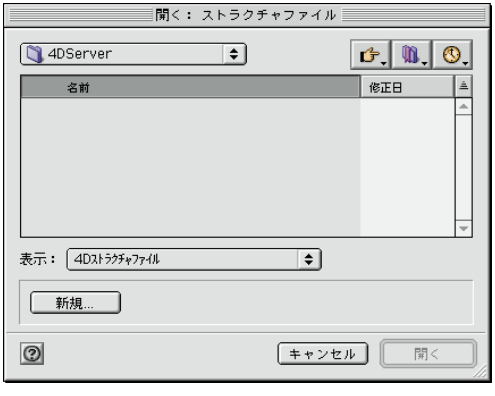
サーバデータベースを作成または開くには、4D Serverを起動します。

1. **4D Serverアイコンをダブルクリックして4D Serverを起動する。**
   Windowsの場合は、4DSERVER.EXEアイコンをダブルクリックします。
   Macintoshの場合は、4D Server 6.5.2アイコンをダブルクリックします。

   「データベースを開く...」ダイアログボックスが表示されるので、既存のデータベースを開くか、新規データベースを作成するかの選択ができます。



Windows版の「データベースを開く」ダイアログボックス

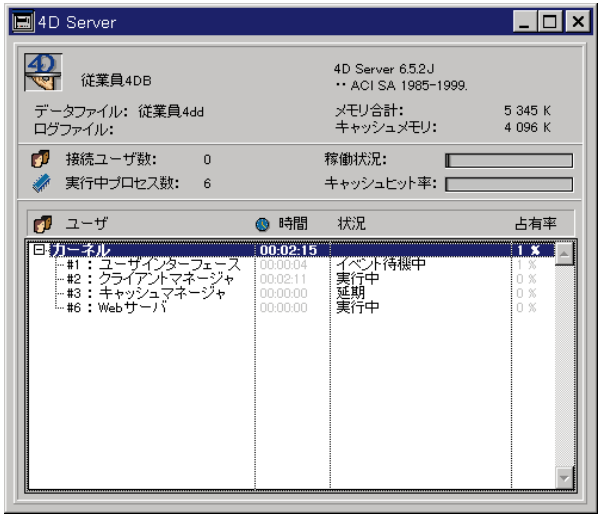


Macintosh版の「データベースを開く」ダイアログボックス

2. **目的のデータベースを選択する。**

3. **「開く」をクリックする。**

4D Serverはデータベースのサービスを開始し、「プロセス」ウインドウを表示します。



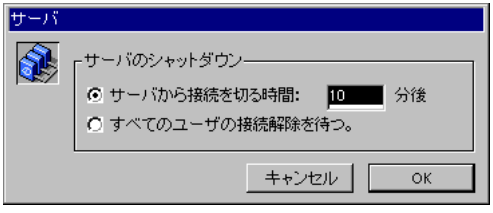
Windows版の「4D Server プロセス」ウインドウ

Macintosh版の「4D Server プロセス」ウインドウ

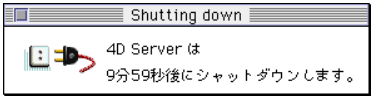
4D Serverの終了

サーバをシャットダウンするには、次の手順に従います。

1. **「ファイル」メニューから「終了」コマンドを選択する。**
「サーバのシャットダウン」ダイアログボックスが表示されます。



2. **何分後にサーバをシャットダウンするかを入力、または“すべてのユーザの接続解除を待つ”オプションを選択する。**
時間を指定して、ダイアログボックスを閉じると、サーバは自動的にクライアントへ終了する旨を伝えます。

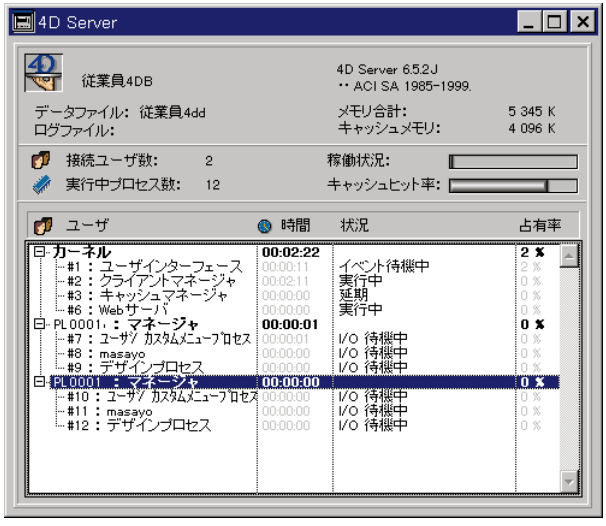


*4D Client*マシン上の「シャットダウン」ウインドウ

このメッセージがシステムに通知されると、新しいクライアントワークステーションはサーバに接続できなくなります。

4D Serverの「プロセス」ウインドウ

4D Serverウインドウには、データベースシステムのさまざまな局面を管理するための情報が表示されます。

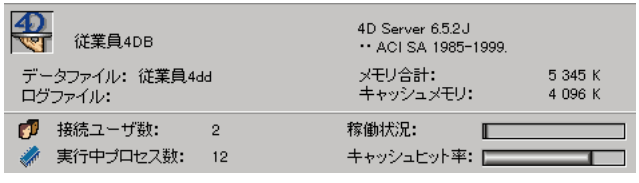


サーバウインドウは、2つの部分に分かれています。

- 上部にはサーバ自体の情報が表示されます。
- 下部には、サーバに接続されているクライアントおよび現在実行中のプロセスについての情報が表示されます。

サーバ情報

「サーバ情報エリア」には、サーバについての一般的な情報が示されます。

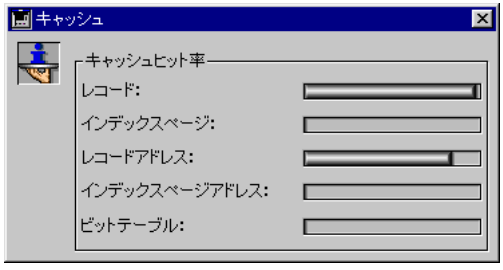


次に示すデータが表示されます。

- **4D Serverのバージョン番号：**現在使用中の4D Serverアプリケーションのバージョン番号。
- **ストラクチャファイル：**4D Serverが開いたストラクチャファイル名。この名前は左上端にある4D Serverアイコンの右側に表示されます。
- **データファイル：**ストラクチャファイルに対応するデータファイル名。
- **ログファイル：**データベース処理を記録するために作成されたログファイル名。ログファイルは、4D Backupプラグインと共に使用することにより、データを修復することができます。ログファイルは4D Backupがインストールされている場合にのみ作成できます。
- **メモリ合計：**4D Serverのカーネルに割り当てられたメモリの合計。Macintoshの場合、これはファインダで4D Server用に確保されたメモリです。Windowsの場合、「データベースプロパティ」ダイアログボックス（4D Serverの）、またはCustomizer Plusユーティリティプログラムを使用してメモリを割り当てます。
- **キャッシュメモリ：**キャッシュに割り当てられたメモリ。キャッシュのサイズにより、メモリに保持できるデータの量を制御し、ディスクへのアクセス回数を減らすことができます。Macintoshでも、Windowsでも、「データベースプロパティ」ダイアログボックス（4D Serverの）、またはCustomizer Plusユーティリティプログラムを使用してキャッシュメモリを割り当てます。
- **接続ユーザ数：**現在接続されているユーザの数。
- **実行プロセス数：**現在実行中のプロセスの数。この数には、すべてのプロセス（カーネル、ユーザ接続、Web接続、ストアードプロシージャ）が含まれます。
- **稼働状況サーモメータ：**現在のサーバの稼働状況を示すサーモメータ（グラフ）。稼働状況サーモメータは、ネットワーク上の全般的な稼働状況を示します。サーバへ送られる要求の量が多くなると、サーモメータの目盛りが長くなります。
- **キャッシュ率サーモメータ：**キャッシュの使用率を示すサーモメータ。サーモメータの目盛りが長いほどキャッシュが多く使われていることを示します。逆にサーモメータが常に短いとキャッシュはあまり使われていないことになります。この場合には、キャッシュのサイズを減らしてもよいでしょう。Macintoshでは、キャッシュのサイズを減らすことは、4D Serverのカーネルルーチンにより多くのメモリが割り当てられます。Windowsでは、キャッシュメモリとカーネルメモリは別々に設定されています。

キャッシュヒット率ウインドウ

「キャッシュヒット率」サーモメータはボタンでもあります。クリックすると、「キャッシュヒット率」ウインドウが表示されます。

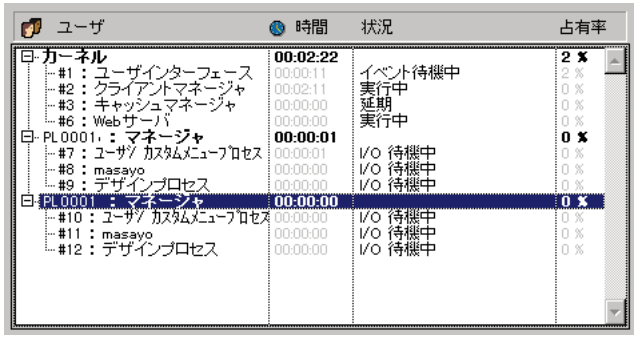


このウインドウにはさらにいくつかのサーモメータがあり、キャッシュの使用状況を示しています。各サーモメータは、キャッシュが扱う各データタイプに対して、どれだけのデータがキャッシュにあるかを示しています。目盛りが長いほど、そのデータタイプを扱う時にディスクアクセス（ヒット）が少なくてすむことを意味し、効率が良いと見なされます。

レコードを使用すると、レコード、レコードアドレス、インデックスページアドレスが連続してキャッシュにロードされます。レコードを追加、変更、削除してデータを変更するとインデックスページとビットテーブルが連続してキャッシュにロードされます。

プロセス情報

「プロセス情報エリア」には、接続中のユーザと現在実行中のプロセスの一覧が表示されます。



リストには、すべてのクライアントプロセスである接続プロセス（ワークステーションローカルのプロセスを除く）が含まれます。ユーザ名の左にある＋印（Windows）または矢印（Macintosh）をクリックすると、カーネルプロセスまたは特定のユーザのプロセスを表示または非表示にすることができます。接続しているクライアントがない場合には、オプションでストアードプロシージャが表示される他には、サーバエンジンの稼働状況だけが表示されます。

4D Serverエンジンは、次に示す4つのカーネルプロセスによって管理されています。

- **ユーザインタフェース：**サーバウインドウ自体のユーザインタフェースを制御します。
- **クライアントマネージャ：**サーバに対するすべての接続を管理し、各クライアントに処理時間を確実に割り振ります。
- **キャッシュマネージャ：**データのディスクへの書き出しを管理します。
- **Webサーバ：**Web接続を管理します。Webサーバプロセスを開始しても、実際のWeb接続を行うのではなく、Webユーザに対してWeb接続の開始を許可するだけです。

サーバウインドウはプロセスごとに次の情報を示します。

- プロセスの名前
- プロセス開始以降消費した時間（秒）
- プロセスのステータス
- 4D Serverがそのプロセスで消費した時間の割合（パーセント）

プロセスを区別するために、「プロセス」ウインドウでは次のようにプロセスを表示します。

- カーネルプロセスは黒
- ユーザ接続プロセスは黒
- Web接続プロセスは青緑
- ストアードプロシージャプロセスは青
- 4D Openベースの接続プロセスは緑

注：アボートされるプロセスは、短い時間ですが、赤で表示されます。

参照

「プロセス」メニュー

パスワード

4D Serverのパスワード管理機能により、アクセス権がユーザに割り当てられます。「パスワード」エディタには、4th Dimensionの「パスワードアクセス」エディタのすべての機能があり、さらに、サーバデータベースへのアクセスを簡素化するオプションも用意されています。

パスドキュメントを作成すると、起動するたびにネットワークとゾーンを選択しなくても、データベースにアクセスすることができます。このパスドキュメントは、「パスワードアクセス」エディタで、ユーザのパスワードと共に保存されます。パスドキュメントにはデータベースの格納場所が保存されます。

パスワードシステムの設定は、4D Clientマシンの「デザイン」モードで次のように行います。

データベースにパスファイルを作成するには、次のようにします。

1. **「ツール」メニューから「パスワード」を選択する。**
   「パスワードアクセス」エディタが表示されます。

2. **パスドキュメントを作成するユーザの名前を反転表示させる。**

3. **「パスワード」メニューから「パス保存...」を選択して、パスを保存する。**
   「パス」ダイアログボックスが表示されます。

4. **パスに名前をつけて「保存」ボタンをクリックする。**

これで、パスドキュメントをダブルクリックするだけでデータベースにアクセスできるようになりました。4D Clientは、自動的にサーバ上のデータベースを探して開きます。

「パスワード」メニューから「パスワードなしでパス保存...」を選択すると、データベースのパスは保存されますが、データベースを起動する時にパスワードを入力する必要があります。

4D Serverデータベースへの接続

4D Clientを起動する場合には、使用しているネットワークコンポーネントに応じて、表示が異なる「接続」ダイアログボックスが表示されます。

「TCP/IP接続」ダイアログボックス

「TCP/IP接続」ダイアログボックスは次の通りです。

Windows版の「TCP/IP接続」ダイアログボックス

Macintosh版の「TCP/IP接続」ダイアログボックス

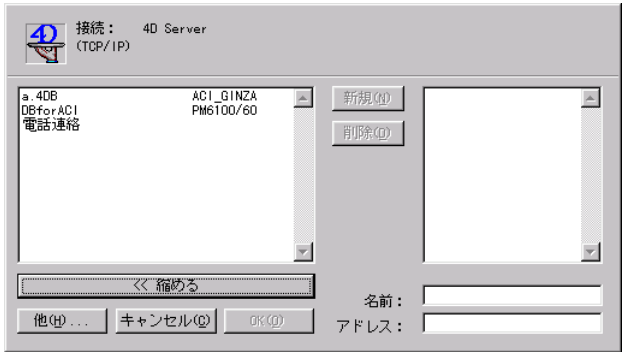
4D Serverには、ネットワーク経由でサーバデータベースの名前を発行する、組み込みのTCP/IPブロードキャスティングシステムがあります。サーバデータベースの名前は、クライアント側の「TCP/IP接続」ダイアログボックスに一覧表示されています。

データベースに接続するには、データベース名をクリックして選択してから「OK」をクリックするか、データベース名をダブルクリックします。「キャンセル」をクリックした場合には、4D Clientは終了し、制御はデスクトップに戻ります。

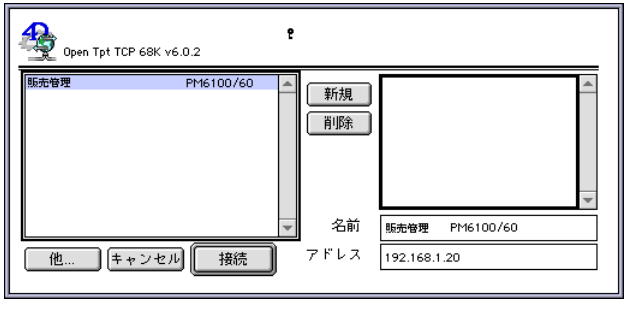
サーバデータベースの名前が自動的にネットワーク上に発行されないように、4D ServerのTCP/IPブロードキャスティングシステムをカスタマイズすることができます。この場合、データベース名はクライアント側の「TCP/IP接続」ダイアログボックスには表示されません。ただし、名前がブロードキャストされないサーバデータベースのIPアドレスがわかっている場合には、手動でIPアドレスを入力してから「OK」をクリックすると、データベースに接続できます。

注：TCP/IPブロードキャスティングの詳細については、『Customizer Plusリファレンスガイド』マニュアルを参照してください。

IPアドレスを手動で入力するには、まず最初に「接続」ダイアログボックスを拡げる必要があります。Windowsの場合は「拡げる」をクリックします。Macintoshの場合は、「接続」ダイアログボックスの右上端にある小さなアイコンをクリックします。



Windows版の「TCP/IP接続」ダイアログボックス



Macintosh版の「TCP/IP接続」ダイアログボックス

サーバデータベースのアドレスを手動で追加するには、「新規」をクリックしてから、エントリに名前をつけてIPアドレスを入力します。ネットワークコンポーネントは「初期設定」ファイルに手動で入力したアドレスを保存します。

アドレスを手動で入力した、あるいは前のセッションで入力したアドレスを選択した後は、「OK」をクリックしてデータベースに接続してください。

「IPX/SPX接続」ダイアログボックス

「IPX/SPX接続」ダイアログボックスは次の通りです。

Windows版の「IPX/SPX接続」ダイアログボックス

Macintosh版の「IPX/SPX接続」ダイアログボックス

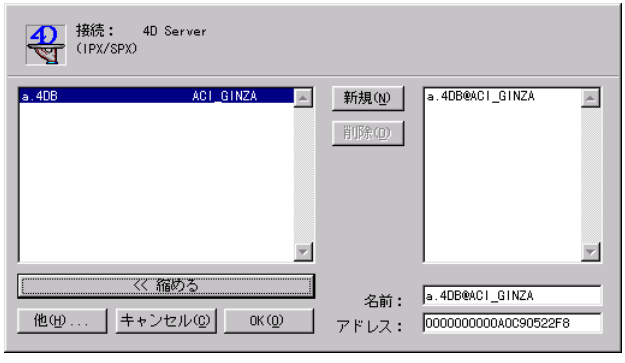
4D Serverには、ネットワーク経由でサーバデータベースの名前を発行する、組み込みのIPX/SPXブロードキャスティングシステムがあります。サーバデータベースの名前は、クライアント側の「IPX/SPX接続」ダイアログボックスに一覧表示されています。

データベースに接続するには、データベース名をクリックして選択してから「OK」をクリックするか、データベース名をダブルクリックします。「キャンセル」をクリックした場合には、4D Clientは終了し、制御はデスクトップに戻ります。

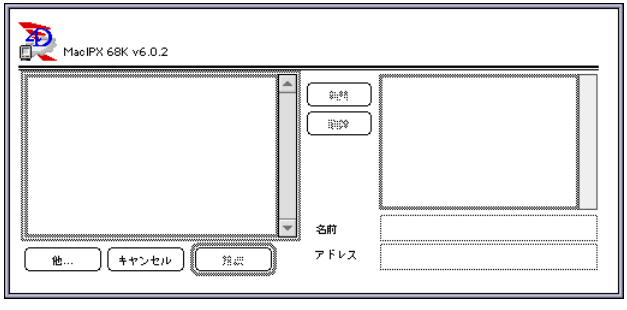
サーバデータベースの名前が自動的にネットワーク上に発行されないように、4D ServerのIPX/SPXブロードキャスティングシステムをカスタマイズすることができます。この場合、データベース名はクライアント側の「IPX/SPX接続」ダイアログボックスには表示されません。ただし、名前がブロードキャストされないサーバデータベースのIPXアドレスがわかっている場合には、手動でIPXアドレスを入力してから「OK」をクリックすると、データベースに接続できます。

注：IPX/SPXブロードキャスティングの詳細については、『Customizer Plusリファレンスガイド』マニュアルを参照してください。

IPXアドレスを手動で入力するには、まず最初に「接続」ダイアログボックスを拡げる必要があります。Windowsの場合は「拡げる」をクリックします。Macintoshの場合は、「接続」ダイアログボックスの右上端にある小さなアイコンをクリックします。



Windows版の「IPX/SPX接続」ダイアログボックス



Macintosh版の「IPX/SPX接続」ダイアログボックス

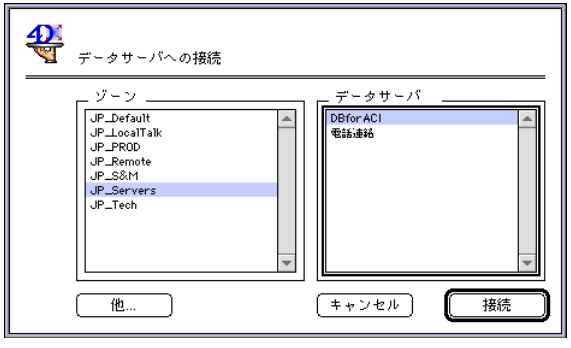
サーバデータベースのアドレスを手動で追加するには、「新規」をクリックしてから、エントリに名前をつけてIPXアドレスを入力します。ネットワークコンポーネントは「初期設定」ファイルに手動で入力したアドレスを保存します。

アドレスを手動で入力した、または前のセッションで入力したアドレスを選択した後は、「OK」をクリックしてデータベースに接続してください。

「AppleTalk 接続」ダイアログボックス

「AppleTalk 接続」ダイアログボックスは次の通りです。

Windows版の「AppleTalk 接続」ダイアログボックス



Macintosh版の「AppleTalk接続」ダイアログボックス

マルチゾーンネットワークの場合には、「AppleTalk接続」ダイアログボックスの左側にはゾーン、右側には選択されたゾーンに発行されたデータベースが一覧表示されます。単一ゾーンのネットワーク（LocalTalk）の場合、「AppleTalk接続」ダイアログボックスには、LocalTalkネットワークに発行されたデータベースだけが一覧表示されます。

サーバデータベースに接続するには、対象のデータベースが発行されているAppleTalkゾーンを選択します。次に、データベース名をクリックして選択してから、「OK」をクリックするか、データベース名をダブルクリックします。「キャンセル」をクリックした場合には、4D Clientは終了し、制御はデスクトップに戻ります。

ネットワークコンポーネントの選択

4D Clientで複数のネットワークコンポーネントが利用可能な場合、各「接続」ダイアログボックスには「他」ボタンがあります。このボタンをクリックすると、「ネットワークコンポーネント接続」ダイアログボックスが表示されます。

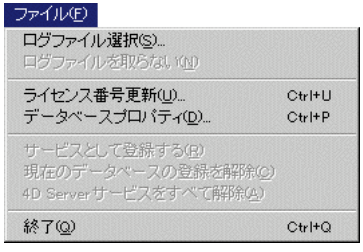
Windows版の「ネットワークコンポーネント接続」ダイアログボックス

Macintosh版の「ネットワークコンポーネント接続」ダイアログボックス

ネットワークコンポーネントを選択するには、コンポーネント名をクリックして選択してから「OK」をクリックするか、コンポーネント名をダブルクリックします。「キャンセル」をクリックした場合には、4D Clientは終了し、制御はデスクトップに戻ります。

ネットワークコンポーネントの詳細については、『4D Serverのネットワークコンポーネント』マニュアルの第1章「はじめに」の「ネットワークプロトコルとネットワークコンポーネント」の節を参照してください。

「ファイル」メニュー

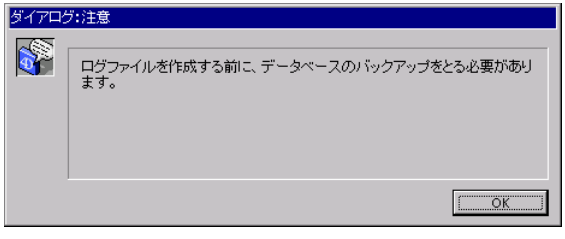


「ログファイル選択...」／「ログファイルをとらない」

4th Dimensionのバックアップユーティリティである4D Backupは、障害が発生する前にデータベースのバックアップを作成することにより、問題を回避します。4D Backupを使用し、データベースのバックアップと、最新のバックアップ後の全変更を記録しているログファイルを作成します。必要に応じてユーザは、レコードが削除されたり、変更されたりする前の状態のデータベースを復元することができます。

注：これらのコマンドを使用するには、4D Serverに4D Backupプラグインがインストールされている必要があります。

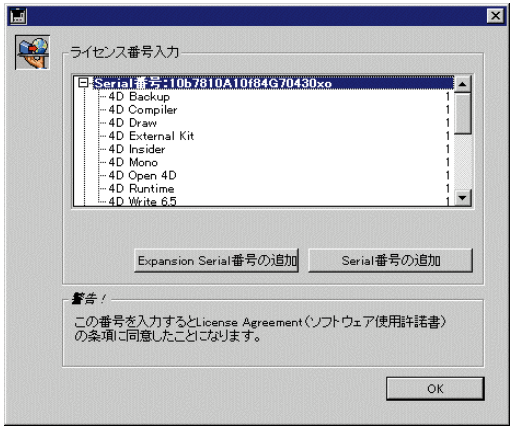
データファイルを最後に変更した後、データベースのバックアップが作成されていない場合、「ログファイル選択...」を選択したときに次の警告が表示されます。



ログファイルへのデータ変更の保存を中止する場合には、「ログファイルをとらない」を選択します。4Dログファイルの詳細については、『4th Dimension ユーザリファレンス』マニュアルを参照してください。

「ライセンス番号更新」

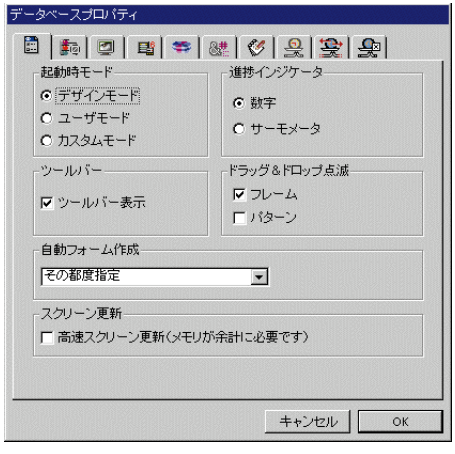
このメニューにより、「ライセンス番号更新」ダイアログボックスが表示されます。



4D Serverには、2つの組み込みクライアント接続が付属します。必要に応じてExpansion Pack（ユーザ数追加パック）または4D Server Internet Expansion（Webライセンス追加パック）をACI製品再販業者から購入して、接続ユーザ数を増やすことができます。ライセンス更新の詳細については、『4D Product Lineインストールガイド』を参照してください。

「データベースプロパティ」

このメニューにより、「データベースプロパティ」ダイアログボックスが表示されます。



「データベースプロパティ」ダイアログボックスを使用して、データベースの一般的なプロパティ（例：サーバデータベースが発行されるTCPポート）を設定できます。「データベースプロパティ」ダイアログボックスの詳細については、『4th Dimension デザインリファレンス』マニュアルを参照してください。

「サービスとして登録」（Windows NT Serverのみ）
「現在のデータベースの登録を解除」
「すべての4D Serverサービスの登録を解除」

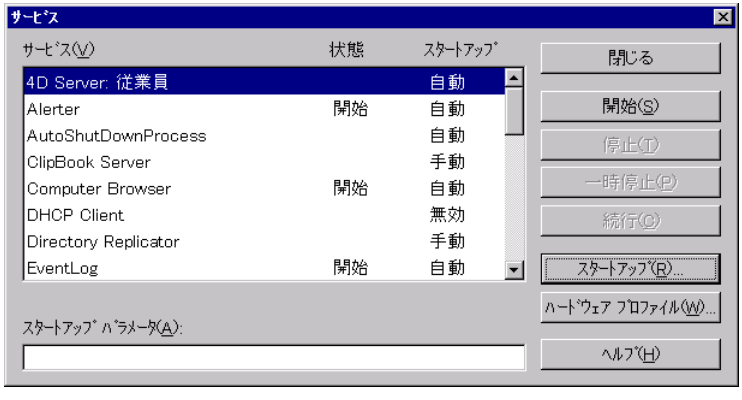
4D Serverは、WindowsNTサービスとして起動できます。

WindowsNTサービスは、サービス制御マネージャが保守するレジストリデータベースにインストールされる実行オブジェクトです。サービスデータベースには、インストールされている各サービスが要求に応じて起動されるのか、または、システムが起動する時に自動的に起動されるのかを決定する情報が含まれています。また、サービスデータベースには、ユーザがログオンしていない時でもサービスが実行できるように、サービス用のログオンと機密保護の情報も納めることができます。また、このデータベースによって、システム管理者は各サービスに対する機密保護要件をカスタマイズできるので、サービスに対するアクセスも制御できるようになります。
一度に実行できるのは、1つのサービスのインスタンスだけです。

4D Serverデータベースをサービスとして登録すると、システムのスタートアップ時に自動的に起動され、ユーザが現在のWindowsログセッションを終了してもシャットダウンされません。

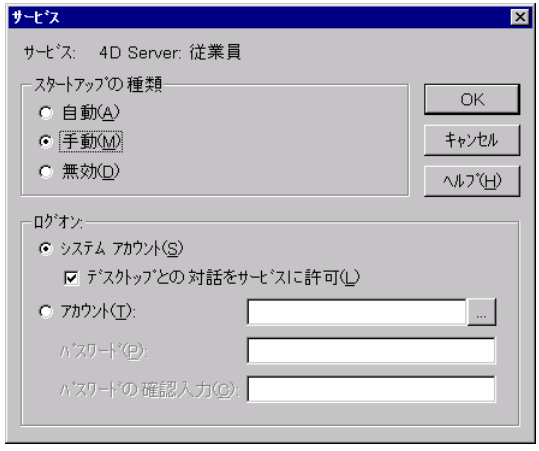
4D ServerデータベースをWindows NT Serviceとして登録するには、「サービスとして登録」を選択します。

ここで、「Windows NT サービスコントロールパネル」を開くと、次のように自分のデータベースの名前が表示されます。



Windows Nサービスとして登録されている4D Serverの“従業員”データベース

データベース名をダブルクリックすると、サービスに対してプロパティを設定できる「サービス」ウインドウが表示されます。



このウインドウを使用して、4D Serverのサービスをシステムスタートアップ時に自動的に起動させる、またはログイン情報を入力することを要求できます。また、4D Serverを（デスクトップとのやり取りをしないように）非表示にすることもできます。

注：データベースはいくつでも登録できます。それぞれのデータベースは一度だけ登録できます。

データベースの登録を解除するには、4D Serverの「ファイル」メニューから「現在のデータベースの登録を解除」を選択します。

4D Serverデータベースの登録を一度にすべて解除するには、4D Serverの「ファイル」メニューから「すべての4D Serverサービスの登録を解除」を選択します。

注：4D Serverのサービスとしての登録は、Windows NT Serverだけで有効です。他のWindowsのバージョンでは、これらの3つのメニューは利用できません。

WindowsNTが起動した時にアプリケーションがサービスとして起動された場合には、4D Serverのサービス登録ステータスは変更できません。このような場合、3つのメニュー項目は選択できません。サービスを停止するには、WindowsNTサービスの「コントロールパネル」を使用してください。

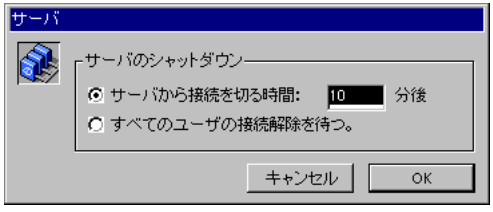
警告：データベースストラクチャファイルのフルパス名は250文字以内です。

「終了」

サーバをシャットダウンする場合には、次の手順に従います。

1. 「ファイル」メニューから「終了」コマンドを選択します。

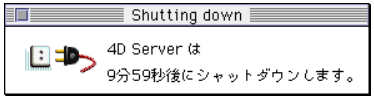
「サーバのシャットダウン」ダイアログボックスが表示されます。



2. サーバを何分後にシャットダウンするかを入力、または“ すべてのユーザの接続解除を待つ ” オプションを選択します。

時間を指定し、ダイアログボックスを閉じると、サーバは自動的にクライアントへ終了する旨を伝えます。

4D Clientマシン上の「シャットダウン」ウインドウ

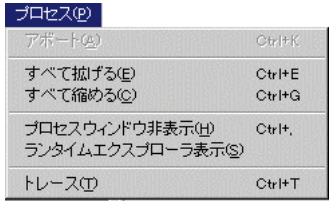


このメッセージがシステムに通知されると、新しいクライアントワークステーションはサーバに接続できなくなります。

参照

「バックアップ」メニュー、「データ」メニュー、「ヘルプ」メニュー、「プロセス」メニュー、「Webサーバ」メニュー

「プロセス」メニュー



「アボート」

警告：「アボート」メニューコマンドは、管理とデバッグのためにだけ提供されています。クライアント自体からプロセスを終了できない場合のみ、このコマンドを使用するようにしてください。

「アボート」コマンドの効果は、「プロセス」ウインドウで現在選択されているプロセスのタイプによって異なります。

- 接続プロセスが選択されている場合、このコマンドはプロセスをアボートします。
- 「ユーザ」リストが選択されている場合、このコマンドはこのユーザに属しているすべてのプロセスをアボートし、リストからそのユーザを削除します。サーバからユーザの接続を自動的に解除します。
- ストアードプロシージャが選択されている場合、このコマンドはストアードプロシージャをアボートします。
- 「ストアードプロシージャ」リストが選択されている場合、このコマンドはすべてのストアードプロシージャをアボートします。
- Web接続プロセスが選択されている場合、このコマンドはWeb接続プロセスをアボートします。
- 「Webクライアント」リストが選択されている場合、このコマンドはWeb接続プロセスをすべてアボートします。

ユーザはカーネルプロセスを除き、すべてのプロセスをアボートできます。カーネルプロセスを選択した場合には、「アボート」コマンドは使用できません。

4Dクライアントマシン上でアボートしたプロセスを使用しようとすると、そのプロセスはもう使用できないという旨の警告メッセージが表示されます。例えば、あるユーザに「アボート」コマンドが適用された場合は次の警告が表示されます。

Error -10001、データベースへの実際の接続は中断されている

注：接続エラーは、『4D Serverのネットワークコンポーネント』マニュアルの「エラーコード」の節に記載されています。

ワークステーションからサーバへの接続が偶然終了した場合、サーバがそのクライアントが予期せず終了したことを認識するのには数秒かかります。この後、サーバは次の事柄を実行します。

- ロックされたレコードあれば解除する
- キャンセルや確定がまだ行われていないトランザクションがあればキャンセルする
- クライアントプロセスをアボートする
- 4D Serverの「プロセス」ウインドウのユーザリストからユーザを削除する

この結果、このユーザに対して「アボート」コマンドを適用する必要はありません。4D Serverが自動的にユーザリストを片付けるためです。

「すべてを拡げる」/「すべてを縮める」

このコマンドは、「プロセス」ウインドウに表示されているすべてのユーザのプロセスを拡げたり、縮めることができます。

「プロセスウインドウ非表示」/「プロセスウインドウ表示」

このコマンドは、「プロセス」ウインドウを表示または非表示にすることができます。

「ランタイムエクスプローラ表示／非表示」

このコマンドは、「ランタイムエクスプローラ」ウインドウを表示または非表示にすることができます。

「トレース」

「トレース」コマンドは、次のプロセスに適用できます。

- プロセス コンテキスト内で「トリガ」の実行をトレースするための接続プロセス
- Webブラウザから送信された要求に応じて、サーバマシン上で実行されたコードをトレースするためのWeb接続プロセス
- サーバマシン上でストアードプロシージャとして実行されたプロジェクトメソッドをトレースするためのストアードプロシージャ

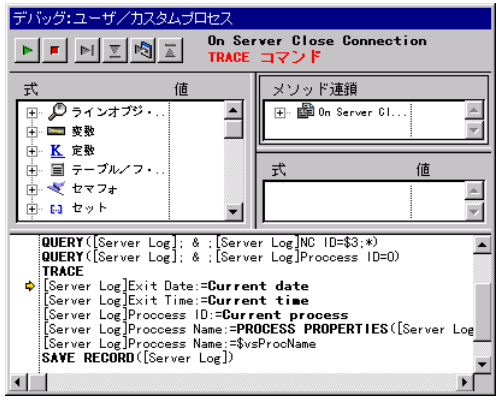
「トレース」コマンドを選択すると、選択したプロセスがコードの実行を開始するとすぐに、そのプロセス用の「デバッグ」ウインドウが表示されます。また、プロセスで実行しているメソッドから**TRACE**コマンドをコールして、そのプロセス用の「デバッグ」ウインドウを表示できます（次の画面例を参照）。

ここで興味深いのは、4D Serverが「トレース」要求を“覚えて”いるという点です。

- プロセスが現在コードを実行している場合には、そのプロセス用の「デバッグ」ウインドウが即座に表示されます。
- 現在プロセスでコードを実行していない場合（つまり、プロセスがデータ入力モードでイベント待ち状態である）、プロセスでコードを再実行するとすぐに「デバッグ」ウインドウが表示されます。

「トレース」コマンドの詳細については、『4th Dimension ランゲージリファレンス』マニュアルの第8章「デバッガ」の「表示されないプロセスやコードを実行していないプロセスのトレース」の節を参照してください。

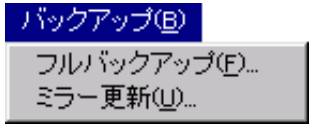
次の「デバッグ」ウインドウでは、サーバマシン上でストアードプロシージャがトレースされています。



参照

「バックアップ」メニュー、「データ」メニュー、「ファイル」メニュー、「ヘルプ」メニュー、「Webサーバ」メニュー

「バックアップ」メニュー



「バックアップ」メニューは4D Backupプラグインがインストールされている場合のみ使用できます。「バックアップ」メニューを使い、データベースのバックアップを管理することができます。詳細については、『4D Backup リファレンスガイド』マニュアルを参照してください。

参照

「データ」メニュー、「ファイル」メニュー、「ヘルプ」メニュー、「プロセス」メニュー、**SELECT LOG FILE**、「Webサーバ」メニュー

「データ」メニュー

「セグメント...」コマンドを選択すると、「セグメント」ダイアログボックスが表示されます。

データベースの作成時、または使用を開始した後にデータファイルを分割することができます。データファイルが大規模になる可能性がある場合は、新しいデータファイルを分割することをお勧めします。データファイルを分割することにより、事実上、無制限にデータを格納することができます。

注：データが2Gバイトを越えない場合や、ハードディスクがデータファイルのサイズに適応できない場合は、データセグメントを作成する必要はありません。

データファイルを分割する時には、データファイルをセグメントに分割してから、各セグメントを格納するボリュームを指定します。例えば、4Gバイトのデータは2Gバイトセグメント2個に分割することができます。セグメントごとにサイズの制限を設けることができるので、他のファイル用にハードディスクの空き領域を確保して、ボリュームをすべて使い切らないようにすることができます。

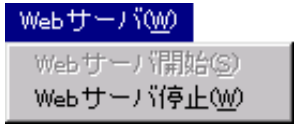
既存のデータファイルのサイズを2Gバイト以上に増やすには、各々2Gバイトまでのデータを格納できるデータセグメントを追加します。サーバマシン上の既存のデータに対してセグメントを作成するには、「セグメント」ダイアログボックスを使用します。

データファイル分割の詳細については、『4th Dimension デザインリファレンス』マニュアルを参照してください。

参照

ADD DATA SEGMENT、「バックアップ」メニュー、「ファイル」メニュー、「ヘルプ」メニュー、「プロセス」メニュー、「Webサーバ」メニュー

「Webサーバ」メニュー



4D Webサービスは、次の3種類の方法で開始することができます。

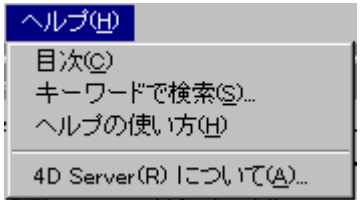
- 4D Serverまたは4th Dimensionの「ユーザ」モードのメインメニューバーから「Webサーバ」メニューを使用する方法。「Webサーバ」メニューを使い、好きなときにWebサービスの開始や停止が行えます。
- データベースがオープンされるたびに、データベースを自動的に公開する方法。Web上にデータベースを自動的に公開するには、4D Serverまたは4th Dimensionの「デザイン」モードで、メインメニューバーの「ファイル」メニューから「データベースプロパティ...」オプションを選択します。すると、「データベースプロパティ」ウインドウが表示されます。「サーバ起動時オプション」より、「起動時にデータベース公開」チェックボックスを選択し、「OK」をクリックします。一度この設定を行うと、4th Dimensionまたは4D Serverでユーザがデータベースを開くたび、自動的にWebに公開されます。
- プログラム上で、**START WEB SERVER**コマンドをコールする方法。

Tips：Web上へのデータベースの公開を開始または停止するために、4Dを終了させてデータベースを再度開く必要はありません。Webサービスの停止と再開は、「Webサーバ」メニューを使用するか、**START WEB SERVER**コマンドと**STOP WEB SERVER**コマンドを呼び出すことにより、何度でも必要なだけ行えます。

参照

「バックアップ」メニュー、「データ」メニュー、「ファイル」メニュー、「ヘルプ」メニュー、「プロセス」メニュー、『4th Dimensionランゲージリファレンス』第61章「Web サービス：システム設定」、『4th Dimensionランゲージリファレンス』第61章「Web サービス：概要」

「ヘルプ」メニュー



「ヘルプ」メニューはWindows上でのみ利用できます。「ヘルプ」メニューを使用し、4D Serverのオンラインヘルプを参照することができます。Windowsヘルプの詳細については、Windowsのマニュアルを参照してください。

参照

「バックアップ」メニュー、「データ」メニュー、「ファイル」メニュー、「プロセス」メニュー、「Webサーバ」メニュー

4D Serverと4D言語

4D Serverバージョン6.5を起動すると、サーバマシン上では次の3つの方法で4Dコードを実行できます。

- トリガ
- ストアードプロシージャ
- データベースメソッド

トリガ

トリガはテーブルに付属するメソッドです。トリガを使用すれば、データベースのレコードに対して「不正な」操作が行われるのを防ぐことができます。トリガは、偶発的にデータが失われたり、変更されたりするのを防ぐだけでなく、テーブルに対する操作を制限するための非常に強力なツールです。例えば、送り状のシステムにおいて、誰かが、送り状の送付先である顧客を指定せずに、送り状を追加するのを防止することができます。

トリガは、データベースエンジンが実際に配置されたマシン上で実行されます。4D Serverでは、トリガはクライアントマシンではなく、サーバマシン上の動作プロセスのコンテキスト内で実行されます。4D Serverではまた、トリガは、データベース操作を起動するユーザプロセスのコンテキスト内で実行されます。しかし、トリガがユーザプロセスのプロセス変数にアクセスすることはありません。

トリガについての詳細は、『4th Dimensionランゲージリファレンス』を参照してください。

ストアードプロシージャ

4D Serverストアードプロシージャは、クライアントマシンではなく、サーバマシン上で実行されるプロセス内でプロセスメソッドを実行するプロジェクトメソッドです。「ストアードプロシージャ」の節を参照してください。

データベースメソッド

次の5つのデータベースメソッドがサーバマシン上で実行されます。

- 「On Server Startup」データベースメソッド
- 「On Server Shutdown」データベースメソッド
- 「On Server Open Connection」データベースメソッド
- 「On Server Close Connection」データベースメソッド
- 「On Web Connection」データベースメソッド

詳細については、それぞれ対応する節を参照してください。

4D Serverと変数

- 4D Serverはインタープロセス変数のテーブルを1つ維持します。これらの変数の有効範囲はサーバマシン内です。コンパイルしたデータベースを実行している場合、インタープロセス変数テーブルの定義は、サーバマシンとすべてのクライアントマシンとで共通です。各クライアントマシンは、それぞれ独自のインスタンスを保持します。
- 各ストアードプロシージャは、それぞれ独自のプロセス変数のテーブルを保持します。コンパイルしたデータベースを実行している場合、プロセス変数テーブルの定義は、すべてのストアードプロシージャと、すべてのクライアントマシン上で実行されるユーザプロセスとで共通です。各プロセスは、それぞれ独自のインスタンスを保持します。
- インタープリタモードにおいて、データベースメソッドとトリガは、実行の各フェーズで動的にプロセス変数を作成し、使用することができます。コンパイラモードでは、これを行うことはできません。コンパイルしたデータベースを実行している場合、データベースメソッドとトリガはプロセス変数の共通のテーブルを1つだけ共有します（このテーブル定義は他のプロセスのものと同じです）。

4D Serverとセット

4D Serverでは、インタープロセスセットとプロセスセットはサーバマシン上に維持されますが、ローカルセットはクライアントマシン上に維持されます。詳細については、「4D Serverとセット」の節を参照してください。

4D Serverとセット

『4th Dimensionランゲージリファレンス』のセットの節で説明しているように、ユーザはインタープロセスセット、プロセスセット、およびローカルセットを使用できます。

- **プロセスセット：**プロセスセットには、それを作成したプロセス内でのみアクセスすることができます。“ UserSet ”と“ LockedSet ”もプロセスセットです。プロセスセットは、プロセスメソッドが終了すると消去されます。プロセスセットは、その名前に特別な接頭辞を必要としません。
- **インタープロセスセット：**名前の前に（<>）記号（小なり記号と大なり記号の組合せ）が付いていれば、そのセットはインタープロセスセットです。

注意：この構文はWindowsでもMacintoshでも使用できます。

- **ローカルセット／クライアントセット：**バージョン6.5では、ローカル／クライアントセットが導入されています。ローカル／クライアントセットの名前の前には、ドル記号（$）を付けます。

4D Serverでは、インタープロセスセットとプロセスセットはサーバマシン上に維持されますが、ローカルセットはクライアントマシン上に維持されます。

Tips：通常は、インタープロセスセットおよびプロセスセットを使用します。これは、クライアント／サーバアーキテクチャにおいて、これらのセットはセット処理を最適化するためです。

4D Serverとセットコマンド

この節では、クライアントマシン上でセットコマンドを実行した場合の動作を、クライアント／サーバアーキテクチャの観点から説明します。

- **CREATE EMPTY SET**
  サーバマシン上に空のセットが作成されます。次に、ネットワークを介して、サーバマシンからクライアントマシンにローカルセットがコピーされます。インタープロセスセットまたはプロセスセットは、サーバマシン上にそのまま維持されます。
- **CREATE SET**
  サーバマシン上にセットが作成されます。次に、ネットワークを介して、サーバマシンからクライアントマシンにローカルセットがコピーされます。インタープロセスセットまたはプロセスセットは、サーバマシン上にそのまま維持されます。
- **USE SET**
  まずローカルセットがネットワークを介して、クライアントマシンからサーバマシンにコピーされ、次にサーバマシン上でそのセットを使用して、テーブルのセレクションが変更されます。インタープロセスセットまたはプロセスセットは、テーブルのセレクションを変更するために、サーバマシン上でローカルに使用されます。

- **SAVE SET**
  ローカルセットは、クライアントマシン上にローカルに保存されます。
  インタープロセスセットまたはプロセスセットは、まずネットワークを介してサーバマシンからクライアントマシンにコピーされ、次にクライアントマシン上にローカルに保存されます。

- **LOAD SET**
  ローカルセットは、ディスクからクライアントマシンにローカルにロードされます。
  インタープロセスセットまたはプロセスセットは、まずディスクからクライアントマシンにローカルにロードされ、次にネットワークを介して、クライアントマシンからサーバマシンにコピーされます。

- **CLEAR SET**

- **ADD TO SET**

- **REMOVE FROM SET**

- **Is in set**

- **Records in set**

上記の5つのコマンドは、クライアントマシン上でローカルにローカルセットにアクセスします。インタープロセスセットまたはプロセスセットでは、情報を入手する、または動作を実行するために、ネットワークを介して要求がサーバマシンに送られます。

- **DIFFERENCE**

- **INTERSECTION**

- **UNION**

上記の3つのコマンドでは、3つのセット引数が同じマシン上に存在する必要があります。したがって、3つの引数すべてがローカルセットであるか、あるいは3つともローカルでないかのどちらかでなければなりません。

- **COPY SET**
  COPY SETを使用して、あらゆるセットを別のセットにコピーできます。例えば、ローカルセットをインタープロセスセットまたはプロセスセットにコピーできます。この場合、セットはネットワークを介して、クライアントマシンからサーバマシンにコピーされます。

4D ServerとUserSet

MODIFY SELECTIONや**DISPLAY SELECTION**フォームでのユーザアクションによるセットの作成を最適化するために、4D Clientは、ローカルセットとして（名前の前に“$”記号は付きませんが）UserSetを作成します。

他の引数がインタープロセスセットまたはプロセスセットである時に、**DIFFERENCE**、**INTERSECTION**、または**UNION**に引数としてUserSetを渡そうとする場合は、まず

UserSet（ローカルセット）をインタープロセスセットまたはプロセスセットにコピーし、そのセットをコマンドで使用してください。

例：

```
ALL RECORDS ([aTable])
 ` ユーザにいくつかのレコードを選択してもらう
MODIFY SELECTION ([aTable];*)
 ` ユーザがレコードを選択したかどうかをチェックする
If (Records in set("UserSet")>0)
   ` 除外するレコードを検索する
  QUERY([aTable];[aTable]aFlag#0)
   ` 結果セレクションからセットを作成する
  CREATE SET([aTable];"To be excluded")
  If (Application type = 4D Client)
     ` UserSetはローカルセットであるため、それをローカルではないセットにコピーする
    COPY SET ("UserSet";"UserSelection") ` <-- ネットワークを介してコピー
     ` DIFFERENCEを呼び出して、3つのローカルでないセット引数を渡す
    DIFFERENCE ("UserSelection";"To be excluded";"UserSelection")
  Else
   ` DIFFERENCEを呼び出す
   DIFFERENCE ("UserSet";"To be excluded";"UserSelection")
  End if
  CLEAR SET("To be excluded")
  USE SET("UserSelection")
  CLEAR SET("UserSelection")
End if
```

下記のコードを使用して、同じことが行えます。

```
ALL RECORDS ([aTable])
 ` ユーザにいくつかのレコードを選択してもらう
MODIFY SELECTION ([aTable];*)
 ` ユーザがレコードを選択したかどうかをチェックする
If (Records in set("UserSet")>0)
   ` 除外するレコードを検索する
  QUERY([aTable];[aTable]aFlag#0)
  If (Application type = 4D Client)
     ` 結果セレクションからローカルセットを作成する
    CREATE SET([aTable];"$To be excluded") ` <--サーバからクライアントにコピー
     ` DIFFERENCEを呼び出して、3つのローカルセット引数を渡す
    DIFFERENCE ("UserSet";"$To be excluded";"UserSet")
```

```
  Else
    ` 結果セレクションからローカルでないセットを作成する
   CREATE SET([aTable];"To be excluded")
    ` DIFFERENCEを呼び出す
   DIFFERENCE ("UserSet";"To be excluded";"UserSelection")
  End if
  CLEAR SET("$To be excluded")
  USE SET("UserSet") ` <--クライアントからサーバにコピー
End if
```

最初の例では、3つのセットが作成され、1つがネットワークを介してコピーされています。
2番目の例では、2つのセットが作成され、2つがネットワークを介してコピーされています。
必要に応じて、上記の例のいずれかを参考にソリューションを選択してください。

4D ServerとLockedSet

“ LockedSet ”は、サーバマシン上で作成され、維持されるプロセスセットです。

参照

4D Serverと4D言語、**COPY SET**、セット

「On Server Startup」データベースメソッド

「On Server Startup」データベースメソッドは、4D Serverでデータベースを開くと、サーバマシン上で一度、呼び出されます。4D Server以外の4D環境では、「On Server Startup」データベースメソッドが起動されることはありません。

「On Server Startup」データベースメソッドは次の事柄を行うには最適です。

- 4D Serverセッション全体を通して使用するインタープロセス変数を初期化する。
- データベースが開かれる時に自動的にストアードプロシージャを開始する。
- 前の4D Serverセッション中に保存された初期設定や各種設定をロードする。
- 明示的に**QUIT 4D**を呼び出すことによって、条件が満たされていない場合に（システムリソースが見つからない等）データベースを開けないようにする。
- データベースが開かれるたびに自動的に実行させたいその他の動作を実行する。

4D Clientがサーバに接続する時に、クライアントマシン上で自動的にコードを実行するには、「On Startup」データベースメソッドを使用してください。

参照

データベースメソッド、「On Server Shutdown」データベースメソッド、SPベースのデータ読み込み（例題）

「On Server Shutdown」データベースメソッド

「On Server Shutdown」データベースメソッドは、4Dをシャットダウンし、データベースを終了する時に、サーバマシン上で一度、呼び出されます。4D Server以外の4D環境では、「On Server Shutdown」データベースメソッドが起動されることはありません。

サーバデータベースは、4D Serverの「ファイル」メニューから「終了」が選択された場合にのみ終了します。

注：ストアードプロシージャでQuit 4Dコマンドを呼び出すと、構文エラーとなり、4D Serverはサーバセッションを続行します。

データベースの終了が開始されると、4Dは次の動作を実行します。

- 「On Server Shutdown」データベースメソッドがない場合、4D Serverは実行中の各プロセスを区別なく1つずつアボートします。
- 「On Server Shutdown」データベースメソッドがある場合、4D Serverは新しく作成されたローカルプロセス内でこのメソッドの実行を開始します。したがって、このデータベースメソッドを使用し、プロセス間通信を介して、他のプロセスに対し、実行を停止するよう通知することができます。結局は、4D Serverが終了するという点に注意してください。「On Server Shutdown」データベースメソッドでは、片付けたり、クローズする操作をすべて実行することができますが、終了を拒否することはできないため、いずれかの時点で終了することになります。

「On Server Shutdown」データベースメソッドは、次の事柄を行うには最適です。

- データベースが開かれた時に自動的に起動されたストアードプロシージャを停止する。
- 次のセッションの始めに「On Server Startup」データベースメソッドで再使用するために、初期設定や各種設定を（ディスク上にローカルに）保存する。
- データベースが終了するたびに自動的に実行させたいその他の動作を実行する。

4D Clientがサーバへの接続を停止する時に、クライアントマシン上で自動的にコードを実行させたい場合には、「On Exit」データベースメソッドを使用してください。

参照

データベースメソッド、メソッド、「On Server Startup」データベースメソッド

「On Server Open Connection」データベースメソッド

「On Server Open Connection」データベースメソッドはいつ呼び出されるか？

「On Server Open Connection」データベースメソッドは、クライアントワークステーションが接続プロセスを開始するたびに、サーバマシン上で一度、呼び出されます。4D Server以外の4D環境では、「On Server Open Connection」データベースメソッドが起動されることはありません。

4D Client

4D Clientでは、「On Server Open Connection」データベースメソッドは次の動作が行われるたびに呼び出されます。

- 4D Clientが接続を行う（「ユーザ」モードプロセスが開始するため）。
- 4D Clientで「デザイン」モードを開く（「デザイン」プロセスが開始するため）。
- 4D Clientで**New Process**コマンドを使用して、非ローカルプロセスを開始する。
- メニューから、または「メソッド実行」ダイアログボックスを使用して、非ローカルプロセスを開始する。

4D Clientでは、いずれの場合にも2つのプロセスが開始されます（クライアントマシン上に1つ、サーバマシン上に1つ）。クライアントマシンでは、プロセスでコードが実行され、4D Serverに要求が送られます。サーバマシンでは、プロセスはクライアントプロセスのためのデータベース環境（ユーザプロセスのためのカレントセレクション等）を管理し、クライアントマシン上で実行中のプロセスから送られた要求に対して応答を返します。サーバ上で実行されるプロセスが「接続プロセス」と呼ばれるのはこのためです。つまり、クライアント／サーバではなく、シングルユーザで実行している場合に、クライアントマシン上で実行されるプロセスが行う処理を、「接続プロセス」は、ネットワーク経由で接続を介し、サーバマシン上で行います。

4D Insider

4D Insiderから4D Serverに接続すると、4D Insiderの作業環境を維持するために、サーバマシン上で接続プロセスが開始されます。このプロセスは、4D Insiderから送られる要求に対して応答を返します。

4D Openベースのアプリケーション

4D Openベースのアプリケーションが4D Serverへの接続を開始するたびに、サーバマシン上で接続プロセスが開始されます。このプロセスは、4D Openを介して送られる要求に対して応答を返し、接続のデータベースコンテキスト（カレントセレクション等）を管理します。

重要：Web接続により、「On Server Open Connection」データベースメソッドは起動されません。Webブラウザが4D Serverに接続する場合は、「On Web Connection」データベースメソッドが起動されます。詳細については、『4th Dimensionランゲージリファレンス』のデータベースメソッドに関する説明を参照してください。

重要：ストアードプロシージャの開始時に、「On Server Open Connection」データベースメソッドは起動されません。ストアードプロシージャはサーバプロセスであり、接続プロセスではありません。ストアードプロシージャはサーバマシン上でコードを実行しますが、4D Client（または他のクライアント）と4D Serverによってやり取りされる要求に対して応答を返すことはありません。

「On Server Open Connection」データベースメソッドはどのように呼び出されるか？

「On Server Open Connection」データベースメソッドは、4D Serverマシン上のこのメソッドを呼び出す接続プロセス内で実行されます。

例えば、4D Clientが4D Serverインタープリタデータベースに接続すると、そのクライアント用に「ユーザモード」プロセスと「デザイン」プロセスが開始されます。したがって、「On Server Open Connection」データベースメソッドは2回実行されます。つまり、1回目は「ユーザモード」接続プロセス内で、2回目は「デザイン」接続プロセス内で実行されます。2つのプロセスがそれぞれサーバマシン上で開始される6番目と7番目のプロセスである場合に、「On Server Open Connection」データベースメソッド内から**Current process**を呼び出すと、**Current process**は1回目には6を、2回目には7を返します。

「On Server Open Connection」データベースメソッドは、サーバマシン上で実行されることに注意してください。このデータベースメソッドは、クライアント側で実行中のプロセスとは無関係に、サーバマシン上で実行中の接続プロセス内で実行されます。また、このメソッドが起動された時点では、接続プロセスにはまだ名前が付いていません（この時点では、**PROCESS PROPERTIES**は接続プロセスの名前を返しません）。

「On Server Open Connection」データベースメソッドは、クライアント側で実行中のプロセスのプロセス変数テーブルにアクセスしません。このテーブルは、サーバマシンではなく、クライアントマシンに常駐します。

インタープリタデータベースでは、「On Server Open Connection」データベースメソッドがプロセス変数にアクセスする場合、このメソッドは接続プロセス用に、動的に作成された専用のプロセス変数テーブルを使用します。「On Server Close Connection」データベースメソッドは結局、同一の接続プロセス内で起動されるので、プロセス変数を使用して2つのメソッド間で情報を維持することも可能です。コンパイラモードでは、こうしたことはできません。
コンパイルしたデータベースでは、「On Server Open Connection」データベースメソッドは、サーバマシン上に保持される共通のプロセス変数テーブルを、トリガや他のデー

タベースメソッドと共用します。このアーキテクチャには2つの目的があります。コンパイルされたコードを実行できるようにすることと、メモリ消費量を削減することです。第一に、ユーザはデータベースメソッドまたはトリガ内から、あらゆるプロセス変数にアクセスできます。プロセス変数は必ず存在する必要があります。第二に、各データベースメソッドまたはトリガに対してそれぞれ1つのプロセステーブルを作成すると、メモリの消費量が増え、初期化に要する時間が増大します。結論として、「On Server Open Connection」データベースメソッドと「On Server Close Connection」データベースメソッドを実行する場合には、プロセス変数に依存しないでください。インタープロセス変数またはテーブルに格納されたデータを使用してください。

4D Serverは、「On Server Open Connection」データベースメソッドに3つの倍長整数タイプの引数を渡し、倍長整数タイプの結果を求めます。したがって、このメソッドでは、3つの引数と関数の結果を倍長整数として明示的に宣言しなくてはなりません。

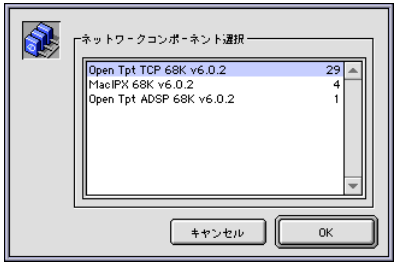
C_LONGINT ($0;$1;$2;$3)

$0に値を返さず、その結果、変数を未定義のままにするか、またはゼロに初期化した場合、4D Serverはデータベースメソッドが接続を受け付けたものとみなします。接続を受け付けない場合には、$0にヌル（null）ではない値を返します。

次の表は、このデータベースメソッドに渡される3つの引数が示す情報を表わしています。

引数	説明
$1	ユーザID番号。4D Serverがユーザを識別するために内部的に使用する。
$2	接続ID番号。各ネットワークコンポーネントが接続を識別するために内部的に使用する。
$3	ネットワークコンポーネントID番号。4D Serverがネットワークコンポーネントを識別するために内部的に使用する。

注：$3で受け取るネットワークコンポーネントID番号は、ユーザが「接続」ダイアログボックスの「他...」ボタンをクリックした時に、クライアント側で表示される「ネットワークコンポーネント選択」ダイアログボックス（下図を参照）に表示される番号です。

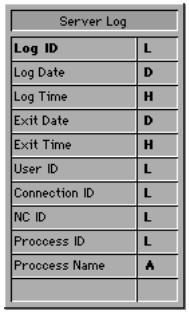


これらの3つのID番号は、例えば4Dコマンドに渡す引数のように、情報ソースとして直接使用することはできません。しかし、これらのID番号は、「On Server Open Connection」

データベースメソッドと「On Server Close Connection」データベースメソッドとの間で、接続プロセスを一意に識別するために利用できます。4D Serverセッションのどの時点でも、これらの3つの値の組み合わせはユニークです。インタープロセス配列またはテーブルにこの情報を格納することによって、2つのデータベースメソッド間で情報をやり取りできます。この節の最後に示された例では、2つのデータベースメソッドが、この情報を使用して、テーブルの同一レコードに接続の開始と終了の日付と時間を格納しています。

例

1. 次の例は、「On Server Open Connection」データベースメソッドと「On Server Close Connection」データベースメソッドを使用して、データベースへの接続ログを管理する方法を示しています。[Server Log]テーブル（下図）は接続プロセスの記録を取るために使用されています。



このテーブルに格納される情報は、次の「On Server Open Connection」データベースメソッドと「On Server Close Connection」データベースメソッドによって管理されます。

```
` 「On Server Open Connection」データベースメソッド
C_LONGINT ($0;$1;$2;$3)
` [Server Log]レコードを作成する
CREATE RECORD([Server Log])
[Server Log]Log ID:=Sequence number ([Server Log])
` 開始の日付と時間（Log DateとLog Time）を保存する
[Server Log]Log Date:=Current date
[Server Log]Log Time:=Current time
` 接続情報を保存する
[Server Log]User ID:=$1
[Server Log]Connection ID:=$2
[Server Log]NC ID:=$3
SAVE RECORD ([Server Log])
` 接続を続行できるようにエラーを返さない
$0:=0
```

` 「On Server Close Connection」**データベースメソッド**
C_LONGINT ($1;$2;$3)
` [Server Log]**レコードを検索する**
QUERY ([Server Log];[Server Log]User ID=$1;*)
QUERY ([Server Log]; & ;[Server Log]Connection ID=$2;*)
QUERY ([Server Log]; & ;[Server Log]NC ID=$3;*)
QUERY ([Server Log]; & ;[Server Log]Process ID=0)
` **終了の日付と時間（**Exit Date**と**Exit Time**）を保存する**
[Server Log]Exit Date:=**Current date**
[Server Log]Exit Time:=**Current time**
` **プロセス情報を保存する**
[Server Log]Process ID:=**Current process**
PROCESS PROPERTIES ([Server Log]Process ID;$vsProcName;$vlProcState;$vlProcTime)
[Server Log]Process Name:=$vsProcName
SAVE RECORD ([Server Log])

Server Log: 4 / 4

Log ID	Log Date	Log Time	Exit Date	Exit Time	User ID	Connection ID	NC ID	Process ID	Process Name
12	97.09.10	17:53:39	97.09.10	18:00:01	18876016	18885864	4	9	ユーザ／カスタムプロセス
13	97.09.10	17:54:22	97.09.10	18:00:00	18876016	18886944	4	8	デザインプロセス
14	97.09.10	18:00:20	00.00.00	00:00:00	14550640	14561672	2	0	
11	97.09.10	17:46:15	97.09.10	17:50:06	14550640	14562752	2	8	デザインプロセス

次に示すのは、[Server Log]テーブルのレコードで、複数の4D Client接続と1つの4D Insider接続を表わしています。

2. 次の例は、午前2時から4時までの間、新しい接続をすべて禁止します。
` 「On Server Open Connection」**データベースメソッド**
C_LONGINT ($0;$1;$2;$3)
If((!!02:00:00!!<=**Current time**)&(**Current time** <!!04:00:00!!))
 $0:=22000
Else
 $0:=0
End if

参照

データベースメソッド、「On Server Close Connection」データベースメソッド

「On Server Close Connection」データベースメソッド

「On Server Close Connection」データベースメソッドは、接続プロセスが終了するたびに、サーバマシン上で一度、呼び出されます。

「On Server Open Connection」データベースメソッドの場合と同じく、4D Serverは「On Server Close Connection」データベースメソッドに 3 つの倍長整数タイプの引数を渡しますが、結果は求めません。

したがって、このメソッドでは3つの引数を倍長整数として明示的に宣言しなくてはなりません。

C_LONGINT ($1;$2;$3)

次の表は、このデータベースメソッドに渡される3つの引数が示す情報を表わしています。

引数	説明
$1	ユーザID番号。4D Serverがユーザを識別するために内部的に使用する。
$2	接続ID番号。各ネットワークコンポーネントが接続を識別するために内部的に使用する。
$3	ネットワークコンポーネントID番号。4D Serverがネットワークコンポーネントを識別するために内部的に使用する。

「On Server Close Connection」データベースメソッドは、「On Server Open Connection」データベースメソッドとちょうど対をなすメソッドです。「接続プロセス」についての詳細は、このデータベースメソッドの説明を参照してください。

例

「On Server Open Connection」データベースメソッドの最初の例を参照してください。

参照

データベースメソッド、「On Server Open Connection」データベースメソッド

ストアードプロシージャ

SQLベースのストアードプロシージャとは？

ストアードプロシージャ（Stored Procedure）という表現は、SQLベースのサーバの世界に由来しています。クライアントワークステーションがSQLベースのサーバに要求を送信する時、実際にはSQLサーバに対してSQL言語で記述されたテキストを送信します。この要求は、実行される前にSQLサーバ上で解析され、解釈されます。要求のソースコードのサイズが大きく、1回のセッション中に要求が何度も送信される場合には、送られる要求の回数が多いほど、ネットワーク経由でソースコードを送信し、解析し、解釈する時間が長くなることは明らかです。

そこで、ネットワーク経由で要求を送信し、解析および解釈を一度だけ行い、クライアントワークステーションから受信するたびにこれを実行する方法を探しました。この解決方法は、要求のソースコード（つまりプロシージャ）をサーバ側に保存し、クライアントワークステーションには実行するプロシージャの名前だけで構成される要求を送らせることでした。結果的に、このプロシージャはサーバ上に「格納（ストア：store）」されるため、「ストアードプロシージャ」という用語になっています。

SQLベースのストアードプロシージャは、クライアントワークステーションから引数を受信し、実現するタスクを（同期的または非同期的に）実行し、最終的に結果をクライアントワークステーションに戻すことができるプロシージャであるということに注意してください。クライアントワークステーションがストアードプロシージャの実行を開始すると、ある程度、サーバマシンにコードの実行を任せます。

4D Serverのストアードプロシージャとは？

4D Serverでは、業界で通用しているストアードプロシージャという名称を使用していますが、4D Serverのストアードプロシージャの機能は、通常のストアードプロシージャの概念をはるかに超えています。

New processのような4Dコマンドを使用すると、メソッドを実行できるユーザプロセスを開始することができます。このメソッドはプロセスメソッドと呼ばれています（『4th Dimension ランゲージリファレンス』の第6章「プロジェクトメソッドとデータベースメソッド」の「プロジェクトメソッド」を参照してください）。

4D Serverを使用し、4D Clientマシン上でも同様の操作が可能です。さらに、4D Serverマシン上で**Execute on server**コマンドを使用すると、メソッドを実行できるユーザプロセスを開始することができます。このメソッドはストアードプロシージャと呼ばれ、（用語の濫用になりますが）サーバマシン上で開始されたプロセスもストアードプロシージャと呼ばれます。

重要：SQLベースのストアードプロシージャと4D Serverのストアードプロシージャの本質的な違いは、SQLベースのストアードプロシージャではSQLプロシージャを実行し、4D Serverのストアードプロシージャでは4Dプロセスを実行するという点にあります。

4Dストアードプロシージャのアーキテクチャ

通常のプロセスと同様に、ストアードプロシージャには次のような独自の環境があります。

- **テーブルごとのカレントセレクション：**各ストアードプロシージャには、個別のカレントセレクションがあります。1つのテーブルは、別々のストアードプロシージャで別々のカレントセレクションを持つことができます。
- **テーブルごとのカレントレコード：**各テーブルは、ストアードプロシージャごとに別のカレントレコードを持つことができます。
- **変数：**各ストアードプロシージャには、独自のプロセス変数があります。プロセス変数は、その変数が設定されたストアードプロシージャの範囲内でのみ認識されます。
- **デフォルトテーブル：**各ストアードプロシージャには、独自のデフォルトテーブルがあります。
- **プロセスセット：**各ストアードプロシージャには、独自のプロセスセットがあります。
- **エラー処理：**各ストアードプロシージャには、独自のエラー処理方法があります。
- **デバッガウィンドウ：**各ストアードプロシージャには、独自のデバッガウィンドウがあります。

ユーザインタフェースの点では、ストアードプロシージャは、ウィンドウを開き、データを表示する（**DISPLAY RECORD**を使用）ことができます。一方、クライアントマシン上のユーザプロセスとは異なり、ストアードプロシージャはデータ入力を開始する（**ADD RECORD**を使用）ことはできません。これは、サーバマシン上にデータ入力カーネルがないためです。

ストアードプロシージャは、システム（ハードウェアおよびメモリ）が許す限りいくつでも開始することができます。事実上、4D Serverマシンは、4D ClientおよびWebブラウザに応答するマシンであるだけではなく、サーバマシンおよび4D Clientマシン上で実行中の他のプロセスと対話するプロセスを実行するマシンである、という見方をする必要があります。

4th Dimensionおよび4D Clientが、ワークステーション上で実行されているユーザプロセスに対してマルチタスク環境を提供しているように、4D Serverは、ストアードプロシージャに対してマルチタスク環境を提供します。たとえば、4D Serverはプロセス間通信用にストアードプロシージャで使用できるインタープロセス変数テーブルを管理しています。

ストアードプロシージャの機能

データ入力を除き、『4th Dimensionランゲージリファレンス』マニュアルで説明されている、ほとんどすべてのプロセスおよびコマンドの機能は、ストアードプロシージャにも適用されます。

ストアードプロシージャでは、レコードの追加、検索、並べ替え、更新、削除が可能です。ストアードプロシージャでは、セットや命名セレクションの使用、ディスク上のドキュメントファイルへのアクセス、BLOBを使用した作業等が行えます。4D Clientマシン上で作業を行う代わりに、サーバマシン上で実行していると考えてください。

ストアードプロシージャの明らかな利点は、データベースエンジンがあるサーバマシン上でローカルに実行されるということです。例えば、ネットワーク経由で**APPLY TO SELECTION**を行うと効率的ではありませんが、ストアードプロシージャ内では効率良く実行されます。「SPベースのデータ読み込み（例題）」の節に示された例では、“ スマート ” なストアードプロシージャを使用して、大幅なパフォーマンスの最適化を実現しています。

しかし、ストアードプロシージャアーキテクチャの最も重要な利点は、4D Serverに新しい世界をもたらすところです。ストアードプロシージャを利用すると、独自の4D Serverサービスを実現することができます。これを制限するのは想像力だけです。「SPベースのデータ読み込み（例題）」の節の例では、4D Serverまたはサーバマシンについての情報をクライアントに提供するストアードプロシージャを示しています。例えば、サーバマシンのボリュームを一覧表示することが可能です。この例は、ディレクトリ情報やドキュメント情報をクライアントに返すように簡単に拡張することができます。

ストアードプロシージャの開始方法

- 4D Clientのユーザモードで「メソッド実行」ダイアログボックスの「4D Server上で実行」オプションを選択すると、ストアードプロシージャを手動で開始することができます。

- さらに、4D Clientで**Execute on server**コマンドを使用して、プログラムからストアードプロシージャを開始することも可能です。
- 4D Server上で実行したメソッド（サーバデータベースメソッドまたはストアードプロシージャ）からは、**Execute on server**または**New process**を使用してストアードプロシージャを開始することができます。

ストアードプロシージャとユーザプロセス間のプロセス間通信についての詳細

ストアードプロシージャ間の通信には、次の方法を使用します。

- インタープロセス変数
- ローカルセマフォまたはグローバルセマフォ
- レコード
- インタープロセスセットおよびインタープロセス命名セレクション
- **GET PROCESS VARIABLE**コマンド、**SET PROCESS VARIABLE**コマンド、**VARIABLE TO VARIABLE**コマンド

『4th Dimensionランゲージリファレンス』マニュアルで、関連する箇所を参照してください。4Dコマンドは、クライアントマシンの範囲内で動作する場合と同様に、サーバマシンの範囲内で動作することに改めて注意してください。

注：CALL PROCESSおよびOutside callの手法は、サーバマシン上では意味がありません。ストアードプロシージャには、データ入力のためのユーザインタフェースがないためです。

さらにもう1つ重要な機能があります。クライアントユーザプロセス（クライアントマシンで実行されるプロセス）は、**GET PROCESS VARIABLE**コマンド、**SET PROCESS VARIABLE**コマンド、**VARIABLE TO VARIABLE**コマンドを使用して、ストアードプロシージャのプロセス変数（*）を読み込んだり、書き込むことができます

（*）サーバマシンのインタープロセス変数と同様です。

重要：GET PROCESS VARIABLEコマンド、SET PROCESS VARIABLEコマンド、VARIABLE TO VARIABLEコマンドを使用して行う、“ マシン間 ” のプロセス通信は、クライアントからサーバに対してのみ可能です。ストアードプロシージャの変数を読み込んだり、書き込んだりするのは常にクライアントのプロセスです。

参照

SPベースのデータ読み込み（例題）、SPベースのサービス（例題）

SPベースのデータ読み込み（例題）

次の例は、クライアントサーバアーキテクチャにおいて、データの読み込みを飛躍的に高速化する方法を示しています。“ Regular Import ”メソッドでは、クライアント側で**IMPORT TEXT**コマンドを使用して、レコードの読み込みに要する時間を調べています。

```
` Regular Importプロジェクトメソッド
$vhDocRef:=Open document ("")
If (OK=1)
  CLOSE DOCUMENT ($vhDocRef)
  INPUT FORM ([Table1];"Import")
  $vhStartTime:=Current time
  IMPORT TEXT ([Table1];Document)
  $vhEndTime:=Current time
  ALERT ("所要時間："+String(0+($vhEndTime-$vhStartTime))+" 秒")
End if
```

通常のデータ読み込みでは、4D Clientはテキストファイルを解析した後、各レコードに対して新しいレコードを作成し、読み込んだデータをフィールドに入力し、レコードをサーバマシンに送信してデータベースに追加します。そのため、大量の要求がネットワーク上でやり取りされることになります。この操作を最適化する方法の1つとして、ストアードプロシージャを使用し、サーバマシンでローカルにこの作業を実行するやり方があります。クライアントマシンではドキュメントファイルをBLOBにロードした後、ストアードプロシージャを開始し、引数としてこのBLOBを渡します。ストアードプロシージャではこのBLOBをサーバマシンのディスク内のドキュメントファイルに保存し、このドキュメントファイルをローカルに読み込みます。ネットワーク要求がほとんどなくなるため、データ読み込みはローカルに（シングルユーザの速度で）処理されます。

次に示すのは、“ CLIENT IMPORT ”プロジェクトメソッドです。このプロジェクトメソッドはクライアントマシンで実行され、後述の “ SERVER IMPORT ”ストアードプロシージャを呼び出します。

```
` CLIENT IMPORTプロジェクトメソッド
` CLIENT IMPORT ( ポインタ ; 文字列 )
` CLIENT IMPORT ( -> [テーブル] ; 入力フォーム )
C_POINTER ($1)
C_STRING (31;$2)
C_TIME ($vhDocRef)
C_BLOB ($vxData)
C_LONGINT (spErrCode)
`データを読み込むドキュメントファイルを選択する
```

```
$vhDocRef:=Open document ("")
If (OK=1)
   `ドキュメントファイルが選択されていた場合には、開いたままにしておかない

  CLOSE DOCUMENT ($vhDocRef)
  $vhStartTime:=Current time
   `メモリにロードしてみる
  DOCUMENT TO BLOB (Document;$vxData)
  If (OK=1)
     `ドキュメントをBLOBにロードできた場合には、
     `サーバマシンにデータを読み込むストアードプロシージャを開始する
     $spProcessID:=Execute on server("SERVER IMPORT";32*1024;"Server Import
Services";Table($1);$2;$vxData)
     `この時点で、このBLOBはこのプロセスでは不要になる
    CLEAR VARIABLE ($vxData)
` ストアードプロシージャが実行する操作が終了するのを待つ
    Repeat
    DELAY PROCESS (Current process;300)
    GET PROCESS VARIABLE ($spProcessID;spErrCode;spErrCode)
    If (Undefined(spErrCode))
       `注：ストアードプロシージャで独自の変数のインスタンスspErrCodeを
       `初期化していない場合、未定義の変数が返されることがある
      spErrCode:=1
    End if
    Until (spErrCode<=0)
     `ストアードプロシージャに対して認識したことを伝える
     spErrCode:=1
    SET PROCESS VARIABLE ($spProcessID;spErrCode;spErrCode)
    $vhEndTime:=Current time
    ALERT ("所要時間："+String(0+($vhEndTime-$vhStartTime))+" 秒")
  Else
    ALERT ("ドキュメントファイルをロードする十分なメモリがありません。")
  End if
End if
```

次は、ストアードプロシージャとして実行される“ SERVER IMPORT ”プロジェクトメソッドです。

```
` SERVER IMPORT プロジェクトメソッド
` SERVER IMPORT ( 倍長整数 ; 文字列 ; BLOB )
` SERVER IMPORT ( テーブル番号 ; 入力フォーム ; 読み込むデータ )
C_LONGINT ($1)
C_STRING (31;$2)
```

```
C_BLOB ($3)
C_LONGINT (spErrCode)
 ` 処理はまだ終了していないため、spErrCodeを1に設定する
spErrCode:=1
$vpTable:=Table ($1)
INPUT FORM ($vpTable->;$2)

$vsDocName:="Import File "+String(1+Random)
DELETE DOCUMENT ($vsDocName)
BLOB TO DOCUMENT ($vsDocName;$3)
IMPORT TEXT ($vpTable->;$vsDocName)
DELETE DOCUMENT ($vsDocName)
 ` 処理が終了したため、spErrCodeを0に設定する
spErrCode:=0
 ` 要求元のクライアントが結果を得るまで待つ
Repeat
  DELAY PROCESS (Current process;1)
Until (spErrCode>0)
```

これら2つのプロジェクトメソッドがデータベースで実行された後で、例えば次のように、“ストアードプロシージャベースの”データ読み込みを実行するように呼び出します。

```
CLIENT IMPORT (->[Table1];"Import")
```

ベンチマークテストを何度か行うと、このメソッドを使用した場合には、通常のデータ読み込みの場合と比べて最高で60倍も速くレコードを読み込めることがわかります。

参照

Execute on server、**GET PROCESS VARIABLE**、**SET PROCESS VARIABLE**、SPベースのサービス（例題）、ストアードプロシージャ

SPベースのサービス（例題）

「SPベースのデータ読み込み（例題）」の節で説明している例では、ストアードプロシージャは、データ読み込み処理が要求されるたびに開始され、終了されています。次の例では、ストアードプロシージャは、サーバデータベースが起動されると自動的に開始され、サーバデータベースに接続している任意の4D Clientから随時に終了する、または再開することができます。ストアードプロシージャは実行されるとすぐに、データベースに接続しているクライアントから送られる複数の要求に対して、非同期的に応答することができるようになります。

「SPベースのデータ読み込み（例題）」の節では、4D Serverで提供される既存のサービスを飛躍的に最適化する方法について説明していますが、この例では、すべての4D Clientで使用できる、新しいサービスやカスタムサービスを実現する方法について説明します。さらに、この例は、独自のサービスを実現するためのテンプレートとしても使用することができます。

ストアードプロシージャを自動起動する

ストアードプロシージャは、「On Server Startup」データベースメソッドによって自動的に開始されます。

`On Server Startup**データベースメソッド**

START SP SERVICES

「On Server Startup」データベースメソッドは、“ SP SERVICES ”プロジェクトメソッドをストアードプロシージャとして開始するため、実際にクライアントがサーバデータベースに接続しているかどうかに関わらず、4D Serverでデータベースが開かれるとすぐに“ SP SERVICES ”が実行されます。次の図では、クライアントがまだ接続していない状態で、ストアードプロシージャが実行されている様子が4D Serverの「プロセス」ウィンドウに表示されています。

ストアードプロシージャを自由に開始・終了する

次に示すのは、“ START SP SERVICES ” プロジェクトメソッドです。

```
` START SP SERVICESプロジェクトメソッド
<>vlSPServices:=Execute on server ("SP SERVICES";32*1024;"SP SERVICES";*)
```

Execute on serverコマンドは、サーバマシン上で呼び出されると**New process**コマンドと同様に動作するため、サーバマシンやクライアントマシンで同じメソッド（START SP SERVICES）を使用して、“ SP SERVICES ” メソッドをサーバマシン上のストアードプロシージャとして自由に開始することができます。

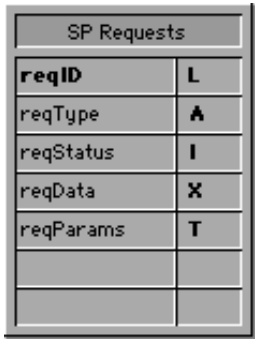
“ STOP SP SERVICES ” プロジェクトメソッドは、“ SP SERVICES ” プロジェクトメソッドに停止するように “ 指示 ” します。

```
` STOP SP SERVICESプロジェクトメソッド
SET PROCESS VARIABLE (<>vlSPServices;vbStopSPServices;True)
```

“ SP SERVICES ” プロジェクトメソッドは、開始されると “ vbStopSPServices ” プロセス変数を “ False ” に設定し、このブール変数が “ True ” になるまでループします。**SET PROCESS VARIABLE**コマンドを使用すると、サーバまたは任意のクライアントマシンで実行されているユーザプロセスは “ vbStopSPServices ” 変数の値を変更でき、結果的にストアードプロシージャを自由に停止できるようになります。

ストアードプロシージャとの通信

ストアードプロシージャは、任意の時間に任意の順序で非同期的にクライアントの要求を受信する、または応答できる必要があります。この通信を保証する簡単な方法は、テーブルを使用する方法です。



[SP Requests]テーブルには、次のフィールドが含まれています。

- [SP Requests]reqIDは、**Sequence number**コマンドを使用して設定されます。このフィールドによって各要求を識別します。
- [SP Requests]reqTypeは、要求のタイプを示します。
- [SP Requests]reqStatusは、次の値のうちいずれかになります。

値	説明
1	要求は送られたが、まだ処理されていない。
0	要求は正常に処理された。
< 0	要求は処理されたが、エラーが発生した。

注：これらの値は、この例題のため任意に選ばれたものであり、4Dから与えられた値ではありません。

- [SP Requests]reqDataは、要求のデータを格納しているBLOBです。要求元から送られたデータ、またはストアードプロシージャから要求元に返されるデータが含まれています。
- [SP Requests]reqParamsには、オプションとして要求元がストアードプロシージャに送った引数の値が含まれています。

なぜテーブルを使用するのか?

クライアントプロセスとストアードプロシージャの間の通信は、**GET PROCESS VARIABLE**、**SET PROCESS VARIABLE**、**VARIABLE TO VARIABLE**コマンドを使用して実現できます。「SPベースのデータ読み込み（例題）」の節や、前述の“ STOP SP SERVICES ”プロジェクトメソッドで使用したソリューションがこの例です。

このような場合には、ストアードプロシージャがさまざまな量のデータを送受信できるようにシステムが設定されていなければなりません。テキスト配列やピクチャ配列等の配列を使用することもできますが、次の2つの理由からテーブルを使用します。

- レコードを使用して要求を処理するアルゴリズムの方が、より容易に実現できます。クライアントマシンから要求を送る処理は、テーブルに要求を追加する処理だけで構成されています。ストアードプロシージャ内から要求に応答する処理は、この要求を修正する処理だけで構成されています。
- 要求はテーブルに格納されるため、ディスク上に保存されます。したがって、要求は（配列に格納されるデータの場合とは異なり）メモリから削除され、要求のサイズが大きい場合でも問題にはなりません。

クライアントマシンから要求を送る

“ Client post request ”プロジェクトメソッドは、要求を送るための汎用的なメソッドです。

` Client post request**プロジェクトメソッド**

` Client post request (**文字列** { ; **テキスト** }) -> **倍長整数**

```
  ` Client post request ( 要求タイプ { ; 引数 } ) -> 要求 ID
CREATE RECORD ([SP Requests])
[SP Requests]reqID:=Sequence number([SP Requests])
[SP Requests]reqType:=$1
[SP Requests]reqStatus:=1

If (Count parameters>=2)
  [SP Requests]reqParams:=$2
End if
SAVE RECORD ([SP Requests])
$0:=[SP Requests]reqID
```

このメソッドから要求ID番号が返されますが、**Sequence number**コマンドを使用することにより、この番号は必ずユニークになります。レコードが[SP Requests]データベースに追加された後、クライアントはフィールド[SP Requests]reqStatusを調べ、ストアードプロシージャが完全に要求を処理するまで待機します。

要求ステータスの検査とクライアントマシンでの結果の取得

“ Client get result ” プロジェクトメソッドは、要求ステータスを調べるための汎用的なメソッドです。前述したように、「SP Requests]reqStatusフィールドが “ 1 ” 以外の値になるとすぐに、クライアントはストアードプロシージャが要求を処理したことが（成功しても失敗しても）分かります。

```
  ` Client get result プロジェクトメソッド
  ` Client get result ( 倍長整数 ; ->BLOB {; 倍長整数 } ) -> 倍長整数
  ` Client get result ( 要求 ID ; ->データ {; 遅延 } ) -> エラーコード
C_LONGINT ($0;$1;$vlDelay])
$0:1)
vlDelay:=0
If (Count parameters>3))
  vlDelay:=$32
End if
READ ONLY ([SP Requests])
Repeat
  QUERY ([SP Requests;=[SP Requests]reqI=$1)
  If (Records in selection([SP Requests]>0)
    If =[SP Requests]reStatus # 1)
      $2->0:=[SP Requests]reDataf
      READ WRITE ([SP Requests])
      While (Locked ([SP Requests]])
        WAITING LOOP ($vlDelay])
```

```
      LOAD  RECORD ([SP Requests])
    End while
    DELETVE RECORD ([SP Requests])
    $0:=[SP Requests]reStatus2
  End if
 Else
 `要求レコードが失われた!
 `あってはならないエラーであるが、エラーを -2に設定（任意の値）
  $0:-$2
 End if

 ` 要求はまだ処理されていない
 If ($0=1)
   WAITING LOOP ($vlDelay)
 End if
Until ($0 # 1)
READ ONLY ([SP Requests])
```

要求がストアードプロシージャにより正常に処理された場合、このメソッドはレコードからBLOBへ結果（ある場合）をコピーします。BLOBへのポインタは引数として渡されます。次に、呼び出し元であるメソッドで要求タイプに応じ、BLOBデータが解析されます。要求の処理が終了したら、クライアントで[SP Requests]レコードの削除を行う必要がある点に注意してください。

次の“ WAITING LOOP ”プロジェクトメソッドは、一定の時間（tick数）が経過するまでループします。

```
` WAITING LOOPプロジェクトメソッド
` WAITING LOOP (倍長整数)
` WAITING LOOP (遅延時間（tick）)
C_LONGINT ($1)
$vlStartTicks:=Tickcount
Repeat
  IDLE
Until ((Tickcount-$vlStartTicks)>=$1)
```

覚書：ユーザモードにおいて、DELAY PROCESSは無効です。“ WAITING LOOP ”プロジェクトメソッドを使用すると、要求がクライアントマシンのユーザモードプロセスから送られた場合でも、プロセスは指定された時間待機します。

ストアードプロシージャとサブルーチン

“ SP SERVICES ” プロジェクトメソッドは、サーバマシン上でストアードプロシージャとして実行されるメソッドです。疑似コードを次に示しますが、総体的なアーキテクチャは簡単です。

```
“ stop ” 変数を初期化する
Repeat
  [SP Requests]reqStatusフィールドが “ 1 ” である要求を検索する
  For 各要求
     要求タイプに応じて、[SP Requests]reqDataフィールドに結果を格納するサブルーチンを呼び出す
    要求ステータスを変更し、クライアントに状況を知らせる
  End for
再度開始する前に少々 “ スリープ ” する
Until “ stop ” 変数が “ true ” になる
```

実際のソースコードは、次のとおりです。

```
  ` SP SERVICESプロジェクトメソッド
  ` ストアードプロシージャ開始
vbStopSPServices:=False
  ` このストアードプロシージャでは、テーブルへのリードライト状態でのアクセスは不要...
READ ONLY (*)
  ` ...[SP Requests]テーブルは例外
READ WRITE ([SP Requests])
Repeat
    ` まだ処理されていない要求を探す
  QUERY ([SP Requests];[SP Requests]reqStatus=1)
    ` これらの要求を一つずつ処理
  For ($vlRecord;1;Records in selection([SP Requests]))
    ` 要求レコードがロックされている場合には、アンロックされるまで待機
    While (Locked ([SP Requests]))
      ` 再試行前に1秒待機
      DELAY PROCESS (Current process;60)
      ` リードライトアクセスを試みる
      LOAD RECORD ([SP Requests])
    End while
    ` 要求が正常に処理されたものとする
    [SP Requests]reqStatus:=0
    Case of
```

```
    \ ([SP Requests]reqType="Server Information")
      SP DO SERVER INFORMATION
    \ ([SP Requests]reqType="Volume List")
      SP DO VOLUME LIST
    \ ([SP Requests]reqType="Browse Directory")
      SP DO BROWSE DIRECTORY ([SP Requests]reqParams)
        ` ...
        ` 他の要求タイプもここに追加可能!
        ` ...
  Else
    ` 要求タイプ不明、エラー " -1 " を戻す（任意の値）
    [SP Requests]reqStatus:=-1
  End case
    ` 要求ステータスが " 1 " 以外の値になるようにする
    ` （サブルーチンによって " 1 " に設定された場合に備えて）
  If ([SP Requests]reqStatus=1)
    [SP Requests]reqStatus:=-3
  End if
    ` 要求レコードを更新
  SAVE RECORD ([SP Requests])
    ` 次の未処理要求に移る
  NEXT RECORD ([SP Requests])
End for

  ` 最後に処理された要求レコードを解放
UNLOAD RECORD ([SP Requests])
  ` 要求に再度応答する前に1秒待機
DELAY PROCESS (Current process;60)
  ` SPが実行を停止するよう指示されるまでループ
Until (vbStopSPServices)
```

" SP SERVICES " プロジェクトメソッドは、データベースに新しいサービスを実現するためのテンプレートとして使用することができます。この節では、" SP DO SERVER INFORMATION " サブルーチンおよび " SP DO VOLUME LIST " サブルーチンの詳細について説明します。" SP DO BROWSE DIRECTORY "（[SP Requests]reqParamsフィールドに納めて送られた引数を引数として取得するサブルーチン）の詳細については、このドキュメントでは説明されていません。

要求のタイプによって、“ SP SERVICES ”プロジェクトメソッドは、結果データを[SP Requests]reqDataフィールドに保存する処理を行うサブルーチンを呼び出します。レコードの保存や、要求ステータスの変更は、“ SP SERVICES ”プロジェクトメソッドによって実行されます。

次に示すのは、“ SP DO SERVER INFORMATION ”サブルーチンです。このサブルーチンは、サーバ関連の情報をBLOBに保存します。別のプロジェクトメソッドを使用して、クライアントマシン上でBLOBデータを取り出します。

` SP DO SERVER INFORMATION**プロジェクトメソッド**

```
TEXT TO BLOB (Application version(*);[SP Requests]reqData;Pascal文字列)
TEXT TO BLOB (Structure file;[SP Requests]reqData;Pascal文字列;*)
TEXT TO BLOB (Data file;[SP Requests]reqData;Pascal文字列;*)
PLATFORM PROPERTIES ($vlPlatform;$vlSystem;$vlMachine)
VARIABLE TO BLOB ($vlPlatform;[SP Requests]reqData;*)
VARIABLE TO BLOB ($vlSystem;[SP Requests]reqData;*)
VARIABLE TO BLOB ($vlMachine;[SP Requests]reqData;*)
```

次に示すのは、“ SP DO VOLUME LIST ”サブルーチンです。このサブルーチンは、ボリューム関連の情報をBLOBに保存します。別のプロジェクトメソッドを使用して、クライアントマシン上でBLOBデータを取り出します。

` SP DO VOLUME LIST**プロジェクトメソッド**

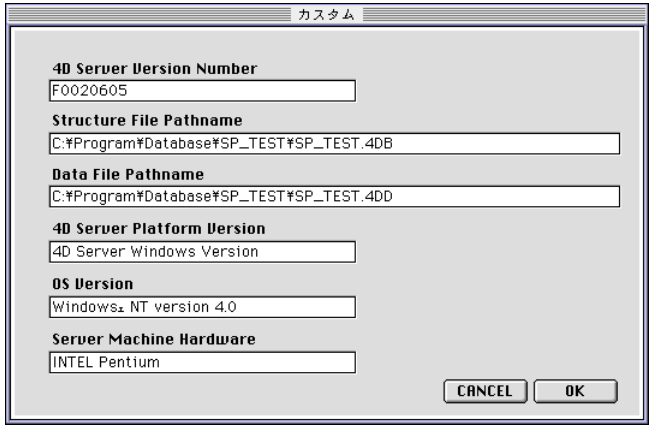
```
VOLUME LIST ($asVName)
$vlSize:=Size of array ($asVName)
ARRAY REAL ($arVSize;$vlSize)
ARRAY REAL ($arVUsedSpace;$vlSize)
ARRAY REAL ($arVFreeSpace;$vlSize)
For ($vlElem;1;$vlSize)
  VOLUME ATTRIBUTES ($asVName{$vlElem};$arVSize{$vlElem};
     $arVUsedSpace{$vlElem};$arVFreeSpace{$vlELem})
End for
VARIABLE TO BLOB ($asVName;[SP Requests]reqData)
VARIABLE TO BLOB ($arVSize;[SP Requests]reqData;*)
VARIABLE TO BLOB ($arVUsedSpace;[SP Requests]reqData;*)
VARIABLE TO BLOB ($arVFreeSpace;[SP Requests]reqData;*)
```

サーバ情報をクライアントマシン上に表示する

汎用的な“ Client post request ”と“ Client get result ”プロジェクトメソッドを使用して、“ M_SERVER_INFORMATION ”プロジェクトメソッドでは、ストアードプロシージャより返されたサーバ情報をクライアントマシン上に表示します。このメソッドは、メニューコマンドに割り当てる、あるいはボタンのオブジェクトメソッドで呼び出してもいいでしょう。

```
` M_SERVER_INFORMATION
C_BLOB (vxData)
C_LONGINT ($vlReqID;$vlErrCode;$vlOffset)
 `要求を送る
$vlReqID:=Client post request ("Server Information")
 `要求ステータスを調べ、結果を取得する
$vlErrCode:=Client get result ($vlReqID;->vxData;60)
 `要求が正常に処理された場合、結果を表示
If ($vlErrCode=0)
  `結果情報をBLOBから取り出す
 $vlOffset:=0
 vsServerVersion:=BLOB to text (vxData;Pascal string;$vlOffset)
 vsStructureFile:=BLOB to text (vxData;Pascal string;$vlOffset)
 vsDataFile:=BLOB to text (vxData;Pascal string;$vlOffset)
 BLOB TO VARIABLE (vxData;$vlPlatform;$vlOffset)
 BLOB TO VARIABLE (vxData;$vlSystem;$vlOffset)
 BLOB TO VARIABLE (vxData;$vlMachine;$vlOffset)
  `プラットフォームプロパティを解析
 vs4DPlatform:="Unknown 4D Server Version"
 vsSystem:="Unknown System Version"
 vsMachine:="Unknown Machine"
 `...
  ` $vlSystemおよび$vlMachineHereを解析するコードをここに記述（記載していない）
  `（PLATFORM PROPERTIESコマンドの例題を参照）
  ` ...
  `結果情報を表示
 DIALOG ([SP Requests];"SERVER INFORMATION")
Else
 ALERT ("要求エラー"+String ($vlErrCode))
End if
`BLOBはもう必要ない
CLEAR VARIABLE (vxData)
```

次の図は、ユーザモードまたはカスタムモードで[SP Requests];"SERVER INFORMATION" フォームを表示したところです。



このウインドウでは、Windows NTベースの4D Serverマシンの情報がMacintoshのクライアントマシン上に表示されています。

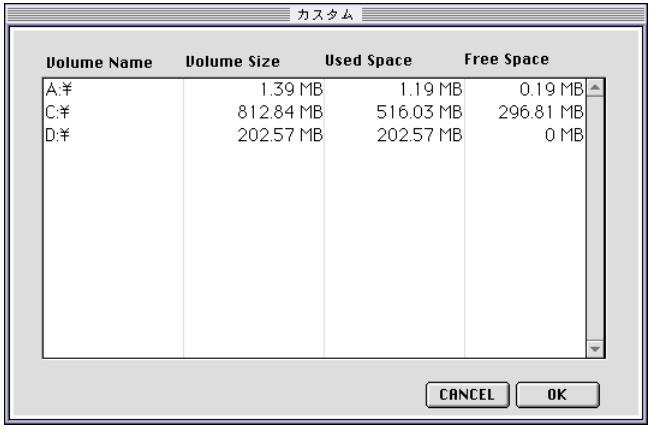
サーバマシンのボリューム一覧をクライアントマシン上に表示する

“ M_SERVER_VOLUMES ” プロジェクトメソッドでは、汎用的な “ Client post request ” と “ Client get result ” プロジェクトメソッドを使用して、ストアードプロシージャより返されたボリューム一覧をクライアントマシン上に表示します。このメソッドは、メニューコマンドに割り当てたり、あるいはボタンのオブジェクトメソッドで呼び出してもいいでしょう。

```
 ` M_SERVER_VOLUMES
C_BLOB (vxData)
 `要求を送る
$vlReqID:=Client post request ("Volume List")
 ` 要求ステータスを調べ、結果を取得する
$vlErrCode:=Client get result ($vlReqID;->vxData;120)
 `要求が正常に処理された場合、結果を表示
If ($vlErrCode=0)
   `結果情報をBLOBから取り出す
  $vlOffset:=0
  BLOB TO VARIABLE (vxData;asVName;$vlOffset)
  BLOB TO VARIABLE (vxData;arVSize;$vlOffset)
  BLOB TO VARIABLE (vxData;arVUsedSpace;$vlOffset)
  BLOB TO VARIABLE (vxData;arVFreeSpace;$vlOffset)
  For ($vlElem;1;Size of array (arVSize))
     `バイト数をMBに変換
```

```
   arVSize{$vlElem}:=arVSize{$vlElem}/1048576
   arVUsedSpace{$vlElem}:=arVUsedSpace{$vlElem}/1048576
   arVFreeSpace{$vlElem}:=arVFreeSpace{$vlElem}/1048576
  End for
`結果情報を表示
  DIALOG ([SP Requests];"VOLUME LIST")
Else
  ALERT ("要求エラー "+String ($vlErrCode))
End if
` BLOBはもう必要ない
CLEAR VARIABLE (vxData)
```

次の図は、ユーザモードまたはカスタムモードで[SP Requests];"VOLUME LIST"フォームを表示したところです。



このウインドウでは、Windowsサーバマシンの情報がMacintoshのクライアントマシン上に表示されています。

参照

BLOBコマンド、**Execute on server**、SPベースのデータ読み込み（例題）、ストアードプロシージャ