Technical Note 04-47
4D for MySQL/PostgreSQL

By Jamras Komoncharoensiri, 4D Evangelist
Technical Note 04-47

（原題: Introduction to 4D for MySQL and 4D for PostgreSQL）

概要

MySQL および PosgreSQL はともに Structured Query Language (SQL)を使用するリレーショナルデータベース管理システム(RDBMS)です。これらのオープンソースデータベースに対して ODBC 経由で接続する 4D ソリューションも存在しますが、今回の Tech Note ではより直接的に通信するための 4D プラグインについて概要を紹介しています。

ビジネスソリューションとしての 4D の利点としては、ラピッド開発環境、スタンドアロンでもクライアント/サーバでも運用できること、Web 機能を備えていることなどを挙げることができます。様々な理由から他のシステム、つまり MySQL や PostgreSQL などを採用する場合があるとしても、バックエンドだけではなく、フロントエンド/ミドルウェアアプリケーションの存在は多くの場合欠かせないものとなっています。

一般にフロントエンドは Java、C、Visual Basic などのローエンド言語で開発し、ミドルウェアには Web サーバがサポートするアプリケーション、大抵は PHP、ASP などのスクリプト言語が採用されます。この種のソリューションは複数のシステムを合わせることによって成り立っています。

強力なソリューションを求めるならば、Web、LAN、ローカルターミナルいずれの方法をとるにしても、共通のゲートウェイを経由して MySQL や PostgresSQL にアクセスするような仕組みが望ましいといえます。そのためにはソフトウェアの種類を最小限にとどめ、管理が簡単で、バックエンドの切り替えが容易であることが求められます。この点で、4D は理想的なフロントエンドであるといえるでしょう。

4D はネイティブコマンドおよびプラグインコマンドとして ODBC アクセスを可能にしているので、直ちに MySQL や PostgreSQL に対して接続することができます。ODBC は汎用性があり、大抵の DBMS に対応しているとはいえ、パフォーマンスの問題やドライバが必須であることなどが短所として指摘されています。4D 2004 の新しいコネクティビティプラグインである 4D for MySQL および 4D for PostgreSQL は、対象 DBMS に対して直接的に通信することを可能にするものです。

4D for MySQL

4D for MySQL プラグインは MySQL API に基づいて設計されており、ODBC レイヤを挟まずに直接 4D から MySQL への通信をするため ODBC ドライバが不要であり、速度面でも最適化されています。実行には 28 種類の専用コマンドが使用されます。INSERT、DELETE、UPDATE、CREATE などの SQL 文は MySQL_Execute コマンドによって実行されます。パラメータは SQL 文の中にハードコーディングするか、タグを使用してダイナミックに指定することができます。

直接ステートメントに値を記述する例

```
$tSQLStatement:="INSERT INTO phone_book VALUES ('Al','Bundy',9547865)"
$ConnID:=MySQL_Connect ($tServer;$tDatabase;$tUsername;$tPassword)
If ($ConnID>0)
	MySQL_Execute ($ConnID;$tSQLStatement)
	If (MySQL_ErrorCode($ConnID)#0)
		ALERT(MySQL_ErrorString($ConnID))
	End if
	MySQL_Close ($ConnID)
End if
$err:=MySQL_LastConnectFailure
```

パラメータでバインドする例

```
$ConnID:=MySQL_Connect ($tServer;$tDatabase;$tUsername;$tPassword)
If ($ConnID>0)
	MySQL_AddStringParameter ($ConnID;"%1";$firstname)
	MySQL_AddStringParameter  ($ConnID;"%2";$lastname)
	MySQL_AddLongIntParameter  ($ConnID;"%3";$phone)
	$tSQLStatement:="INSERT INTO phone_book VALUES "+    $tSQLStatement:=$tSQLStatement+"
(%1,%2,%3)"
	MySQL_Execute ($ConnID;$tSQLStatement)
	If (MySQL_ErrorCode($ConnID)#0)
		ALERT(MySQL_ErrorString($ConnID))
	End if
	MySQL_Close ($ConnID)
End if
$err:=MySQL_LastConnectFailure
```

BLOB やピクチャを扱う場合、後者の手法は必須です。
SELECT 文の場合は MySQL_Execute の代わりに MySQL_Select を使用します。SELECT 文が正しく実行されると 1 以上の参照番号が返されます。エラーメッセージは MySQL_ErrorString で取得することができます。SELECT 文の結果はコマンド

MySQL_Get<Type>Field で取得することができますが、このコマンドは一度に 1 行のレコードしか返さないので、セレクション全体を読み取るためにはループが必要です。以下はループでセレクションを読み取るコードの例です。

```
While (MySQL_NextRow ($selectID)=1)
` Call one of the MySQL_Get<Type>Field command
` to obtain the current column value into a variable
End while
MySQL_CloseSelect ($selectID)
```

4D for PostgreSQL

4D for MySQL と同様、ODBC を介さずに直接通信することのできる 46 種類のローレベルプラグインコマンドが用意されています。SELECT 文だけが別コマンドとなっている 4D for MySQL とは異なり、すべてのステートメントが PGSQL_Execute コマンドで実行されます。コマンドの種類を見分けるためには返り値を参照し、PGRES_COMMAND_OK であれば SELECT 以外のコマンドであるとみなされます。PGRES_TUPLES_OK であれば、コマンドは SELECT であり、コマンド PGSQL_Get<Type>Value による結果の処理が求められます。

SELECT 以外のステートメント

```
$ConnID:=PGSQL_Connect($tServer;$port;$option;$Tty;$tDBName;$tUName;$tPswd)
If ($ConnID>0)
	$resID:=PGSQL_Execute($ConnID;$tSQLStatement)
	If ($resID#0)
		$err:=PGSQL_GetResultStatus($resID;$resultStatus)
		If ($resultStatus#"PGRES_COMMAND_OK")
			` Execution fail
			ALERT($resultStatus)
		End if	PGSQL_CloseResult($resID)
	End if
	PGSQL_Close($ConnID)
Else
	ALERT(PGSQL_GetLastConnError($ConnID))
End if

$tSQLStatement:=”INSERT INTO phone_book VALUES ('Al','Bundy',9547865)”
$tSQLStatement:=”DELETE FROM phone_book WHERE firstname=’Al’, lastname=’Bundy’, phone=95
47865”
$tSQLStatement:=”UPDATE phone_book SET firstname=’Peg’ WHERE lastname=’Bundy”

$ConnID:= PGSQL_Connect($tServer;$port;$option;$Tty;$tDBName;$tUName;$tPswd)

If ($ConnID>0)
	$resID:= PGSQL_Execute($ConnID;$tSQLStatement)
	If ($resID#0)
	$err:= PGSQL_GetResultStatus($resID;$resultStatus)
	If ($resultStatus#"PGRES_TUPLES_OK")
		` Successfully executed
	` Handle the result
	Else
```

```
            ` Execution fail
            ALERT($resultStatus)
        End if
        PGSQL_CloseResult($resID)
End if
        PGSQL_Close($ConnID)
Else
        ALERT( PGSQL_GetLastConnError($ConnID))
End if
```

SELECT ステートメント

```
$tSQLStatement:="SELECT firstname, lastname, phone FROM phone_book”
$ConnID:= PGSQL_Connect($tServer;$port;$option;$Tty;$tDBName;$tUName;$tPswd)
If ($ConnID>0)
        $resID:= PGSQL_Execute($ConnID;$tSQLStatement)
        If ($resID#0)
            $err:= PGSQL_GetResultStatus($resID;$resultStatus)
            If ($resultStatus="PGRES_TUPLES_OK")
                $err:= PGSQL_GetRowCount($resID;$rowcount)
                For ($row;0;$rowcount-1)
                    $err:= PGSQL_GetTextValue($resID;0;$row;$fname)
                    $err:= PGSQL_GetTextValue($resID;1;$row;$lname)
                    $err:= PGSQL_GetLongintValue($resID;2;$row;$phone)
                    APPEND TO ARRAY(arrfname;$fname)
                    APPEND TO ARRAY(arrlname;$lname)
                    APPEND TO ARRAY(arrphone;$phone)
                End for
            Else
                ` Execution fail
                ALERT($resultStatus)
            End if
            PGSQL_CloseResult($resID)
        End if
        PGSQL_Close($ConnID)
Else
        ALERT( PGSQL_GetLastConnError($ConnID))
End if
```