



[プライマネージ]

PRIMANAGE

CSS Open Package Series for Enterprise Resource Planning

データベーステーブルへの項目追加手順

DB(データベース) に アクセス する方式

PRIMANAGE2007(あるいは PRIMANAGE MEISTER)で、データベースの追加(サブシステムの追加)、テーブルの追加、テーブル内の項目の追加、等を行おうとすると、クライアント(第1相:GUI)あるいは業務処理(第2相:業務処理)から、データベースにアクセスするときに、どのような動作を経て、アクセスが完了するのを知っている必要があります。

【補足】

PRIMANAGE2007では、第1相、第2相、第3相の各相が、相互に互いに独立になっているので、各相間の相互のやり取りが、(慣れるまでは)直感的に把握しにくい面があります。

また、PRIMANAGE2007を、コードの面から見た場合は、サブシステム間も相互に独立になっています。(サブシステム相互のデータの交換はデータベースを通じて行われ、コードは互いに独立しています。これにより、新しいサブシステムを既存のシステムに組み込む場合に、他のサブシステムのコードに触れずに組み込みを行うことができます。)

しかし、各サブシステムが相互に共通に利用している部分もあります。このような部分は、基本的には1つなので、新規にサブシステムを追加するような場合は、既存の共通部分に、新規のサブシステムの情報を組み込むことも必要になります。

データベースへアクセスする機能は、この相互に共通に利用している部分に該当します。

データベースに新しいサブシステムを追加したり、あるいは既存のサブシステムに属するDBにテーブルを追加する場合、どのようなところでどのような変更を加える必要があるのかを、データベースへのアクセスを行うときに発生する動作を通じて調べます。

【注】 PRIMANAGE2007では、「DB アクセス関数」という言い方をよく行います。これは文字通り、「DB にアクセスする関数」、ということですが、実際には、2種類のタイプの関数が、同じ用語で呼ばれます。

1つは、DBOpen() のように、第1相、あるいは第2相で使用され、あるいはDBInsert() のように、第2相で使用されるもので、DBにどのような操作をするのかを関数名から知ることができるタイプのものであります。

このタイプの関数は、先頭に“DB”がつき、具体的には、

DBOpen()、DBCLOSE()、DBFetch()、DBInsert()、DBUpdate()、DBDelete()、
DBCommit()、DBRollback()、および DBSel[n]() ([n]は0から始まる整数を表します。
例えば、DBSel0()、DBSel1()、…)

しかありません。

もう一方のタイプの関数は、埋め込み型のSQL文を埋め込んだ関数で、DBの各テーブルに一つずつ作成されるものです。

このタイプの関数は、DBエンジンと直接のやり取りを行うものです。

このタイプの関数は、第3相だけに現われ、関数名の先頭に“Dxxx”がつきます。

(xxxは、サブシステムのコード) このタイプのソースコードは、(Oracleの場合は) .pc という拡張子がつきます。

PRIMANAGE MEISTER の例では、DMESToku.pc がこのタイプに属します。

前者のDBアクセス関数(例: DBOpen())は、結局、後者のDBアクセス関数を使って、実際の処理を行うのと、出現する相が異なるので、慣れると混乱は起こらないのですが、このドキュメントに関しては、

前者を、「一般DBアクセス関数」、後者を単に「DBアクセス関数」と呼ぶことにします。

なお、説明を具体的にするために、

- ① 部品プログラムからアクセスをしようとしている、DBのテーブルを“ttt”、
- ② tttへのアクセスのしかたについての指定(“LOCK/UNLOCK”の指定と、“読み出し専用”などのモードについての指定、を合わせたもの)を“ppp”、
- ③ サーバー側で起動される、tttへのアクセスを扱うための、サブシステム単位のモジュールを“mmm”、
- ④ 各テーブルに対応しているDBアクセス関数を“DBaf_t()”、

で表わします。

1. 一般 DB アクセス関数が呼ばれた場合の動作(クライアントからの呼出し)

クライアント側で PRIMANAGE2007 (あるいは PRIMANAGE MEISTER) の部品を実行しているときに、一般 DB アクセス関数(例:DBOpen()) が呼ばれた場合、次のような動作が順に発生します。

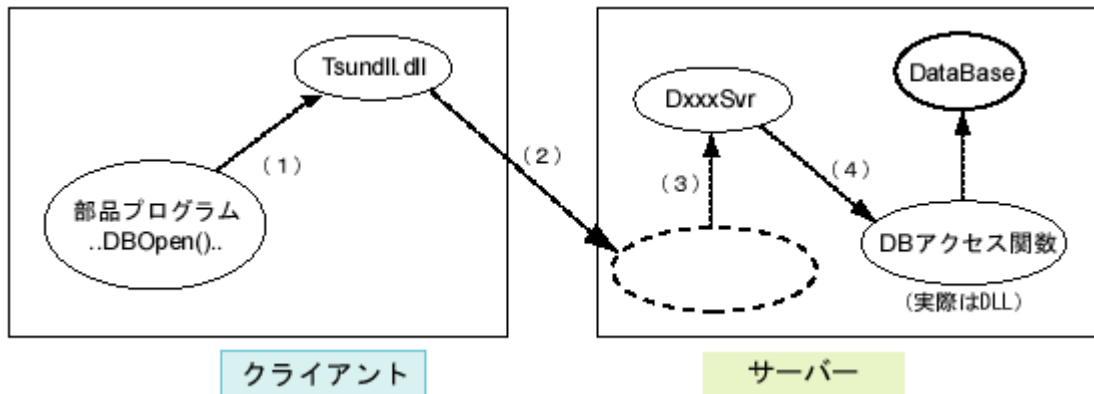


図 1

(1) tsundll.dll 中の関数の起動。

→ 一般 DB アクセス関数は tsundll.dll に実装されているため。例えば、DBOpen() が呼ばれた場合、tsundll.dll 中の DBOpen() が呼ばれます。

→ なお、一般 DB アクセス関数を呼び出すときには、パラメータとして、① 対象となるテーブル、② テーブルを ロックするかどうか、③ モード(読み出し専用、読み書き両用、…)、を与えます。

【注 1】 対象となるテーブルは、そのテーブルの ID(C_symbol.h で定義された 6~8 桁の数字)で指定します。

【注 2】 このように、DB にアクセスしようとするとき、“DB のテーブル”と“操作”を指定するだけで、処理は第 2 相以降に任せます。このように第 1 相は、処理の具体的内容からは独立しています。

(2) (tsundll.dll 中の)関数での処理。

→ (tsundll.dll 中の)一般 DB アクセス関数は、処理を遂行を サーバーに依頼します。依頼の内容は、

(a) (一般 DB アクセス関数は、) ttt という DB テーブルに対して、

(b) ppp という処理を行いたいのので、

(c) mmm というモジュールを立ち上げて(起動して)、処理を委任して欲しい。

(d) なお、この mmm に、パラメータ yyy^[注 3] を渡して欲しい。

というものです。

mmm は、DxxxSvr.exe という形のファイル名 (xxx はサブシステム名) をした関数です。

(つまり簡単に言えば、「処理の実際は mmm という関数が行うので、その関数に処理を任せて、結果をこちらに戻して下さい」というものです。)

【注 3】 yyy は、実際には、一般 DB アクセス関数に渡された ppp を変換したのになります。例えば、一般 DB アクセス関数 DBOpen() が呼ばれたとき、ppp として、「テーブルを UNLOCK」、「読み出し専用でオープンする」、を指定されている場合、DBOpen() は、これを PR_OPEN_R というパラメータに変換して、これを yyy として mmm に渡します。

→ tsundll.dll 中の一般 DB アクセス関数は、実行の過程で サーバーへ処理の遂行を依頼するときに、サーバーとの通信を行う必要があります。(DxxxSvr.exe は、サーバー側にいます。)

このとき、メッセージの長さを知るために、対象としているテーブルのサイズ(バイト数)が必要になります。

このサイズを知るための関数の名称(DBGetSize_usr())は、すでに一般 DB アクセス関数の中に組み込まれており、一般 DB アクセス関数は、テーブルのサイズを知る必要が生じると、この関数を自動的に呼び出します。

(新規にサブシステムを作成して、新規にテーブルを作成した場合や、新規のテーブルをサブシステムに追加した場合は、この関数の呼び出しに回答する部分がありませんから、この呼び出しに回答する部分を 実装する必要があります。)

→[1]

→ また、一般 DB アクセス関数は、サーバーで起動を行うモジュールの名称(mmm)も、(サーバー側に教える)必要があります。

このとき、サーバーで起動するモジュールは、サブシステム単位で動作するものです。(つまり、アクセスのリクエストの対象のテーブル名が異なっても、それらのテーブルが 同じサブシステムに属している場合は、同じモジュール:mmm を起動します。)

これは、一般 DB アクセス関数は、リクエストの対象となっているテーブルが、どのサブシステムの、どのテーブルであろうとも、全部取り扱うのに対して、実際のテーブルはサブシステム単位で管理されているため、この一般 DB アクセス関数の内部で、アクセスの対象となっているテーブルが、どのサブシステムに属しているのかを判定する必要があることを表します。

一般 DB アクセス関数の内部には、サーバーで起動を行うモジュールの名称を取得する関数の名称(DBGetName_usr())が組み込まれており、一般 DB アクセス関数は、サーバーで起動を行うモジュールの名称を知る必要が生じると、この関数を自動的に呼び出します。

(この場合も、新規にサブシステムを作成して、新規にテーブルを作成した場合や、新規のテーブルをサブシステムに追加した場合は、この関数の呼び出しに回答する部分がありませんから、この呼び出しに 回答する部分を実装する必要があります。)

→[2]

なお、サーバー側で起動するモジュール(mmm)は、DxxxSvr.exe という形式をしています。
(xxx は、サブシステム・コード)

(例: PRIMANAGE MEISTER のテーブル: MES_TOK_TBL に対してアクセスをする場合、起動するのは、(PRIMANAGE MEISTER のサブシステム・コードは MES ですから、) DMESSvr.exe となります。)

→ 以上の準備が整ったところで、一般 DB アクセス関数は、サーバーと通信を開始します。

(3)サーバー側での処理。

→ サーバー側は、(クライアント: より正確には、クライアント側にいる、一般 DB アクセス関数)からの 送信を受信すると、その求めに応じて、クライアント側から指定されているモジュール: mmm(=DxxxSvr.exe) を 起動します。

(4)指定されたモジュールでの処理。

→ クライアント側からの指定で起動されたモジュール(mmm)は、クライアントが求めている DB テーブル(ttt)に、やはりクライアントが求めている操作(ppp。パラメータ yyy に変換されている)を 実行することになります。

具体的には、モジュール(mmm)は、ttt に対応する DB アクセス関数(DBaf_t())を識別し、この関数に(ppp と等価な)処理パラメータ yyy を渡して、この関数を起動します。

DBaf_t()は、DB のエンジン部とやり取りを行い、ttt に対して、リクエストされている操作を行います。

(この操作には、埋め込み型の SQL 文を用います。)

→ モジュール(mmm)は、サブシステム単位で起動されますので、どの DBaf_t()を起動するべきなのかは、あらかじめ知ることはできず、起動のときに渡される処理用のパラメータを見て、初めて知るようになっています。

これは、第 3 相は、他の相(第 1 相 および 第 2 相)から、独立しているためです。

第 1 相、第 2 相が、第 3 相に求めることは、対象となっているテーブル(ttt)と、行うべき処理の指定(ppp)を与えたときに、その結果を戻してもらうことだけです。

つまり、第 1 相 あるいは第 2 相は、第 3 相で、DB アクセス関数(Dbaf_t()) の名称や、パラメータの種類と渡し方、あるいは実装について、一切知りません。(=関知しません。別の表現のしかたをすると、「何も知らないでも 動作することができるようになっていきます」。)

→ モジュール(mmm)の実装では、実行の過程で DB アクセス関数(Dbaf_t())へのアクセスを実行する 段階になると、特定の関数 SSGetfunc_usr()を呼び出すようになっています。

(関数名を取り出すような名前ですが、実際はこの呼出しによって、DB アクセス関数を起動します。そして、DB アクセス関数の実行結果を受け取ります。)

この関数 SSGetfunc_usr()は、その内部で、関数: SCxxxGetfunc() を呼び出し、SCxxxGetfunc()は、自分のサブシステムに属する DB アクセス関数のうちから、リクエストに対応する DB アクセス関数(Dbaf_t) を 選択して実行するように動作します。

つまり、モジュール(mmm)の実装では、

1. 関数 SSGetfunc_usr()の実体を組み込み、
2. SSGetfunc_usr()の実体の中で、SCxxxGetfunc() を呼び出すようにして、
3. この関数を通じて目的とする DB アクセス関数を呼び出すように、関数: SCxxxGetfunc() を作成するようする必要があります。

(この結果、SSGetfunc_usr()が呼び出されると、リクエストに対応する Dbaf_t()が選り出され、処理パラメータ が渡されて、Dbaf_t() が実行されるようになります。)

→[3]

【注 4】まわりくどい説明ですが、DB アクセス関数の起動は、モジュール(mmm)の中にある、固定の名称の関数: SSGetfunc_usr() から行われます。

これに対して、実際に起動すべき DB アクセス関数は、サブシステム単位で管理されているので、まずどのサブシステムについてのリクエストなのかを判定します。

次いで、サブシステム単位で作成する関数を通じて(この段階で、対象となる DB アクセス関数の存在する 範囲が狭められています)、リクエストされている処理を行う DB アクセス関数を起動する、という順序になっています。

→ また、モジュール(mmm)でも、データの送信を行います^[注 5]。このため、メッセージの長さを算出することが必要になり、テーブル ttt のサイズを知る必要が起こってきます。

【注 5】 Dbaf_t() を実行した結果を戻す必要があります。

テーブル ttt のサイズを知るために、モジュール(mmm)では、固定の名称の関数: SSGetRsize_usr()を呼び出しますので、この関数の実体を実装する必要があります。

→[4]

以上が、一般 DB アクセス関数(例:DBOpen()) が呼ばれた場合に生じる、一連の動作の説明です。(正確には、DB アクセス関数の起動まで。)

これから、新規にサブシステムを作成した場合、あるいは既存のサブシステムに新しくDBのテーブルを追加すると、DBへのアクセスという面から見た場合、4つの関数と、補足の関数2つ、を実装する必要となります。

これを表の形でまとめると、以下のようになります。

[特記事項]

新しいサブシステムを、既存のシステムに追加する場合は、上記の[1]～[4]項の情報を提供できるようにする必要があります。実際には、上記の[1]～[4]項のそれぞれについて、それぞれの情報が(実行時に)必要になった場合に、それぞれ特定の関数が呼ばれますので、その関数を実装するだけで、(実行に必要な)情報を提供できるようになっています。具体的には:

	呼ばれる関数名	目的	動作	ソースファイルの名称
[1]	DBGetRsize_usr ()	対象としているテーブルのサイズ(バイト数)を得る	① → 5	SCUSRbra.c
[2]	DBGetPname_usr ()	サーバーで起動を行うモジュールの名称を得る	②	
[3]	SSGetfunc_usr()	テーブルに対応するDBアクセス関数の名称を得る(実際には、DBアクセス関数が呼び出される)	③ → 6	SSUSRbra.c
[4]	SSGetRsize_usr ()	対象としているテーブルのサイズ(バイト数)を得る	① → 5	
以下は、サブシステム毎に異なる名前となる(xxxにサブシステム・コードが入る。例: SCMesGetRsize())				
5	SCxxxGetRsize ()	サブシステムの範疇で、テーブルのサイズを返す	---	SCUSRbra.c
6	SCxxxGetfunc()	サブシステムの範疇で、テーブルのアクセス関数を呼び出す	---	SCxxxtsu.c

① 対象となっているテーブルが属するサブシステムを判定して、そのサブシステムのSCxxxGetRsize()を呼び出す。実際のサイズの算出は、SCxxxGetRsize()が行う。

② 対象となっているテーブルが属するサブシステムを判定して、C_symbol.h に定義されている DxxxSvr の名を返す。(マクロ定義されている。)

③ 実際には、SCGetxxxfunc() を呼び出します。関数の名称は “GetPname” となっているので、あたかも、プログラムの名称を知るのが目的であるように見えますが、実際には SCGetxxxfunc() を呼び出しますので、DB アクセス関数 (Dbaf_t()) が起動されます。

2. 一般 DB アクセス関数が呼ばれた場合の動作 (サーバーからの呼出し)

1. では、クライアント側からの、一般 DB アクセス関数 (例: DBOpen()) が呼ばれた場合の説明をしましたが、サーバー側からの呼出しの場合も、同じ動作をします。

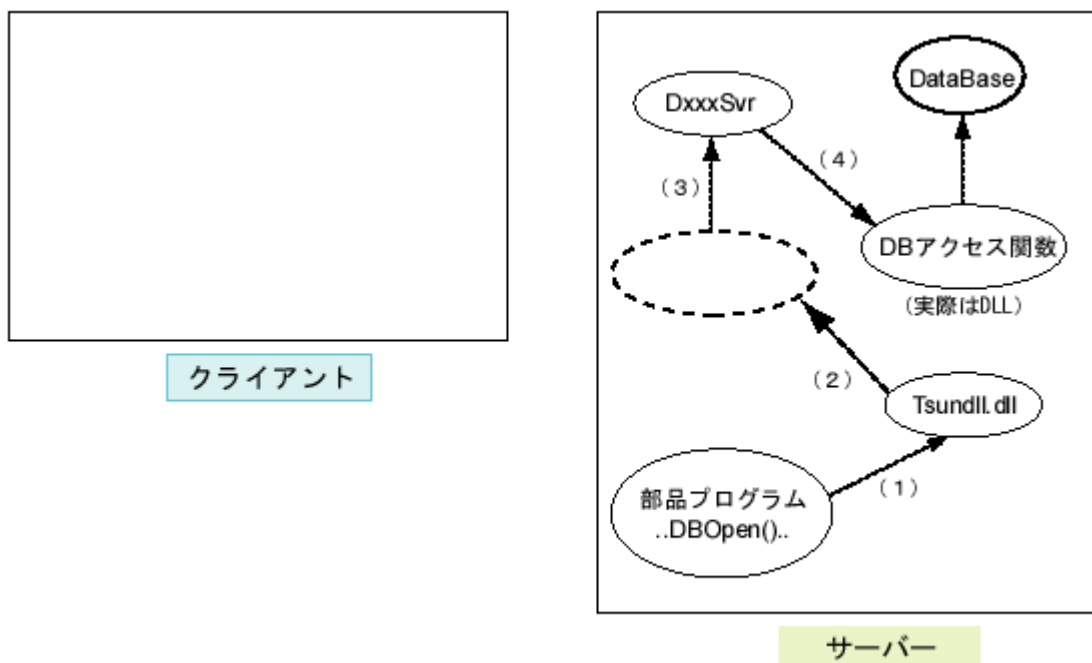


図 2

この場合、一般アクセス関数の呼出しで実行されるコードは、サーバー側にある tsundll.dll のコードです。

一般アクセス関数の呼び出しで、呼び出し元が クライアント側にいるか、サーバー側にいるか、で最も大きく異なる点は、クライアント側からは、DB のテーブルの内容を変更する一般 DB アクセス関数^{【注6】}の呼び出しを全く行わないことです。

【注 6】 DBInsert()、DBUpdate()、DBDelete()。従って DBCommit()、DBRollback() も含まれます。

これは、機能的にできないのではなく、PRIMANAGE2007 では、DB の内容を変更するのは、第 2 相 (業務処理) から行う、という原則を守っているためです。

なお、第 1 相から DB にアクセスすることができるのは、GUI で滑らかな操作を行えるようにするためです。

(例：担当者コードを入力したときに、その担当者の名前や所属が GUI 画面に表示されると、作業をしている人は、入力したコードの適否を直ちに確認できます。また、データがないような場合も、直ちに次の操作に移ることができます。これは DB に保存されているデータを読み出しているのですが、このような場合の DB アクセスを第 2 相を通じて行くと、サーバー側に負荷をかけ、応答も遅くなるので、このような場合には DB に直接アクセスしています。)