＜ 会計計算を Stored Procedure で行う＞

Stored Procedure はデータベース上で、アプリケーションからの指示を受け取って結果だけをクライアントに返します。あるいは結果を返さない場合もありますが、結果はテーブルあるいはビューとして返されます。

Stored Procedure を使用する利点は
①サーバー上で必要なデータをもとにレコードや結果の計算などを返すのでネットワークトラフィックが最低限ですむこと
②一度データベースに記述しておけばどのアプリケーションからもいつでも呼び出せること
③アプリケーションで同じ作業をするよりもパフォーマンスがよいこと
などです。
窓口での会計計算はアプリケーション上では集計クエリーの結果として求めることも出来ますが、会計計算はまさに Stored Procedure を使用するためにあるような仕事なので Stored Procedure を使用することにします。
ただし Stored Procedure を使って会計計算をするには＜データベース内で完結的に結果を得ることが可能であること＞という条件がつきます。アプリケーションで使用するデータベースが 1 個だけからなる単一データベースアプリケーションであればこのような問題を考える余地はありません。
しかし入力されたデータの保持をするデータベースとマスターを提供するデータベースが別のファイルであれば、考慮に入れておかねばなりません。プロフェッショナルドクターはまさにデータの入力用のデータベースと、参照用のマスターを収めるデータベースが別々のデータベースになっていますので、この点をあらかじめ考慮に入れています。具体的には、カルテに処方などのデータを入力する際に、マスターの点数をそのままデータのテーブルにコピーしています。この時点で一つのデータベースのデータが別のデータベースのデータに移動します。移動にかかる時間はほとんどゼロです。このようにしているために会計計算時に改めてマスターを呼び出す必要はありません。同時にこれにより Stored Procedure での会計計算が可能になります。また点数改定時にそのまま計算を旧マスター点数のままで行うことが出来ます。

Stored Procedure はProcedure 言語で記述します。
Stored Procedure には入力パラメータ・出力パラメーターの記述が可能です。
たとえば、特定のカルテ番号の診療レコードの点数の合計を求める、という場合にはカルテ番号が入力パラメーターとして必要です。

診療点数の合計を求める Stored Procedure は以下のようになります。この結果は必ず 1 行で返されます。

```
------------------------------------------------------------------
CREATE PROCEDURE GET_SINRYOUTEN_GOUKEI(KARTE_NO INTEGER)
RETURNS (sinryouten_goukei INTEGER) AS
BEGIN
SELECT SUM(TEN*DAYS*RYOU)
FROM KARTE_SINRYOU
WHERE KARTE_NO=:karte_no
INTO :sinryouten_goukei;
END
------------------------------------------------------------------
```

1 行目の CREATE PROCEDURE GET_SINRYOUTEN_GOUKEI(KARTE_NO INTEGER)の(KARTE_NO INTEGER)が入力パラメーターです。
また 2 行目の RETURNS (sinryouten_goukei INTEGER) AS の(sinryouten_goukei INTEGER)が出力パラメーターです。

Stored Procedure を作成したら、実際に機能を確認してみます。
カルテ番号 10116 から診療点数の合計を求めてみます。

```
------------------------------------------------------------------
```

データベースの<KARTE_SINRYOU>テーブルには以下のようなレコードが入力されているとします。

NUMBER	KARTE_NO	CODE	RYOU	DAYS	TEKIYOU_CODE	FUTAN_KUBUN	TEN
27	10124	111000470	1	1	<null>	H	130
25	10117	140039050	1	1	<null>	H	290
29	10116	112000970	1	1	<null>	H	35
28	10116	112000210	1	2	<null>	H	74
26	10123	150205810	1	1	<null>	H	5060
18	10115	112700310	1	1	<null>	H	74
23	10115	180016410	1	1	<null>	H	53

```
------------------------------------------------------------------
```

診療点数の合計を求めるためにデータベース上で以下のように SQL を入力します。

```
execute GET_SINRYOUTEN_GOUKEI 10116;
```

10116 は特定のカルテ番号を示す入力パラメーターです。

結果は以下のようになります。なお、データが Null の場合は Null が返されます。
アプリケーション上では Null の場合は結果は 0 となります。

```
SINRYOUTEN_GOUKEI
=================
              183
```

＜Stored Procedure のアプリケーションでの利用＞

この Stored Procedure をアプリケーションで利用するには以下のようにします。
①アプリケーションのフォーム上に StoredProcedure オブジェクトを配置します。
②DataBase プロパティを設定します。（DBDATA など）
③StoredProcName プロパティでデータベース上にある Stored Procedure を選択します。
この場合、StoredProcName プロパティは GET_SINRYOUTEN_GOUKEI です

このように設定して Params プロパティを表示させると、パラメーターとして

```
0-KARTE_NO
1-SINRYOUTEN_GOUKEI
```

と表示されています。
このうち 0-KARTE_NO が入力パラメーターであり、
1-　SINRYOUTEN_GOUKEI が出力パラメーターということになります。0、1 はパラメーターインデックスであり、KARTE_NO、SINRYOUTEN_GOUKEI はパラメーター名です。
クエリー・Stored Procedure などのパラメータを使用するデータセットはあらかじめパラメーターのデータ型がわかっていないと使用することが出来ません。クエリーの場合は手動でパラメーターの型を設定しなければなりませんが、Stored Procedure の場合は配置した StoredProcedure コンポーネントを一度 Active にすれば自動的にパラメーター値・パラメーターの型がパラメーター表に記入されます。

このようにプロパティを用意しておいてから、次に実際に Stored Procedure を使用するために順序として、
①0-KARTE_NO（入力パラメーター）を設定し、
②Stored Procedure を実行する
③1-SINRYOUTEN_GOUKEI（出力パラメーター）を得る
という一連の作業が必要です。Stored Procedure の結果は出力パラメーターとして与えられます。

Stored Procedure を結果をアプリケーションで実験的に得るために実際に必要なコードは以下のようになります。

```
procedure Tform1.Button4Click(Sender: TObject);
begin
SP_GET_SINRYOUTEN_GOUKEI.Params[0].Value:=StrToInt('10116');
SP_GET_SINRYOUTEN_GOUKEI.ExecProc;
Label15.Caption:=IntToStr(SP_GET_SINRYOUTEN_GOUKEI.Params[1].Value);
end;
```

<パラメーター名>.Params[0].Value が Stored Procedure のパラメーター値にアクセスする構文です。
簡単のために 10116 というカルテ番号をそのまま設定していますが、実際にはフォーム上の特定のコントロール値を参照します。また結果自体も会計結果テーブルに入力します。
この procedure を実行すると、
Label15 のラベルに 183 と表示されます。
これで会計計算の結果を Stored Procedure で求め→アプリケーションのコントロールに入力するという一連の操作が可能になります。