Professional Doctor C/S は Delphi Ver5 で開発されています。Delphi は優れたアーキテクチャーに基づく RAD ツールとして大きな信頼性を持っています。

Delphi は Windows 上での開発ツールであることが唯一の欠点といわれてきたのですが、近々 Linux 上への移植が完了するはずで、Linux 上でも優れた開発環境として No.1 の地位を獲得するでしょう。これで欠点のないツールに進化できるわけです。

Professional Doctor C/S は今後データベースを Linux 上に置くだけではなく、アプリケーションを Linux 上で稼動させることも当然視野においています。そのためには Delphi で開発することが最短距離です。
Delphi で開発することのメリットは、
データベースアプリケーションをも開発できること。
開発が短期間で可能なこと。
ツールのバージョンアップ後でもほとんどそっくりそのままコードが使用できること。
Delphi は何よりも限界を知らないツールであること。があげられます。

今回から連載で Professional Doctor C/S の開発経過を掲載します。
必ずしも内容は一貫性・連続性はなく、特定のトピックについての読み切りとします。この点をご了承ください。

電子カルテの開発にはデータベースの処理に関する手法のほとんどすべての問題が絡みますし、単純な処理ばかりではありません。したがって、他の医療関係データベースアプリケーションの開発にも参考になるものと思われます。
但し、掲載記事についてのお問いあわせには応じられませんのでご了承ください。

患者名簿の情報を検索する

はじめに患者名簿の情報を検索する＜ID 検索フォーム＞について解説します。
患者の名前はわかっているが ID がわからない、電話番号をもとに住所を表示させたい、あるいは同じ名前の患者を全員表示する、というようなときに使用する患者検索フォームが必要です。単にデータを参照するだけの機能ですので単純ですが、1 つのフォームを作りコンポーネントをいくつか置いてそのプロパテイの設定をして、特定のボタンに短いコードを記述することだけで簡単に出来てしまうということを見ていただきたいと思います。

データベースデータの検索には 2 つの方法があります。
①一つは先頭から検索語句に一致する内容のデータを探し、合致する項目にカーソルを移動する。
②もう一つは SQL を使用して検索内容に一致するレコードを SQL の結果として表示する。

SQL ベースの C/S データベースでは患者名簿データの全員分を一気に表示することはありません。
他のデータと同様、患者名簿は常に一つのパラメータを与えられたときに必要なデータ（通常 1 名分）のみをサーバーから取得し、クライアントコンピュータに結果を表示することになります。したがって患者名簿の内容を横断的に検索するための手段として患者名簿テーブルに対する問い合わせを実行することが必要になります。
このため＜ID 検索フォーム＞の場合はパラメーターケリーの結果を表示するという②の方法を取っています。
この場合は検索条件に一致するレコードを、複数あれば複数レコードを一気に表示します。
検索条件はパラメータークエリーのパラメーターとして与えられます。
検索条件としてのパラメーターが変わればそれに応じた SQL 文が必要になりますので、パラメーターに応じた数の SQL 文を用意することが必要になります。

＜ID 検索フォーム＞の概要は以下の通りです。

目的：入力した検索のためのデータをもとに患者名簿の他の情報を検索・表示させるためのフォームです。
機能：ID ・電話番号・名前・名前のふりがな・住所のいずれかの情報をもとに患者のデータを検索します。検索結果はフォーム内のグリッドに表示されます。
条件：ID ・電話番号はデータに入力されている情報の通りに入力する必要があります。
名前・ふりがな・住所はその一部分の入力で可です。
検索画面の使用方法は次のようにします。ID で検索したい場合は ID という見出しのついたラジオボタンをクリックしたのち、＜②検索語句を入力します＞と表示されているコントロールボックスに ID 番号を入力します。次に＜③検索実行＞ボタンをクリックすると下段のグリッドに問い合わせの結果レコードが表示されます。

実装
フォームの名前を＜IDKarteNoForm＞とします。
このフォームのユニット名を保存します。
TQuery の名前を＜QKensaku＞
TDataSource の名前を＜KensakuDS＞
にします。
＜ID 検索フォーム＞に必要なオブジェクトは

TRadioGroup：RadioGroup1
TQuery：QKensaku
TDataSource：KensakuDS
DBGrid：DBGrid1

TButton
です。

RadioGroup 内にラジオボタンを作成するときは RadioGroup 内にラジオボタンをドラッグするのではなく、RadioGroup の Items プロパテイ内にラジオボタン名を記述していきます。これによりラジオボタンが作成されます。このようにして作成したラジオボタンにはアクセスが可能になります。（どのラジオボタンがチェックされたかは Checked プロパテイで知る必要がありますのでラジオボタンに対してアクセスが必要です）

QKensaku にはプロパテイとしての SQL 文は入力してありません。かわりに検索を実行するときに、検索手段によって必要な SQL 文を動的に作成します。タイミングは検索ボタンをクリックしたときです。
名前・名前のふりがな・住所をもとに検索するときは検索内容の一部の入力しかないときにも検索可能にするために正規表現による問い合わせが必要になります。

DBGrid1 の DataSource プロパテイは KensakuDS であり、KensakuDS の DataSet プロパテイは QKensaku です。このようにして DBGrid1 内に QKensaku の実行結果を表示することができます。なお DBGrid は特定の DataSource の連結する DataSet のデータを表示しますが、機能に融通性がありデータセットと緩やかに結合することが可能で、その結果表示する項目を動的に変更して表示することができます。DBGrid1 の DataSource プロパテイさえ決めておけば DataSource のもつデータセットがどのようなものに変更されてもその内容を表示することが出来ます。ただし、メモ型項目は表示できません。
このフォームで実行された結果の住所は表示できないのはそのためです。
DBGrid 内で表示されないメモ型項目を表示するには別の処理の追加が必要ですが、ここでは触れません。

＜③検索実行＞ボタンをクリック時のコードは以下のようになります。

```
procedure TIDKarteNoForm.KensakuButtonClick(Sender: TObject);
Var AtaiOfID:Integer;
AtaiOfPhone:String;
AtaiOfName:String;
AtaiOfFurigana:String;
AtaiOfJuusho:String;

begin
```

```
if EditKensakuNaiyou.Text='' then
exit;

if   IDRadioButton.Checked=True then
 With QKensaku  Do begin                          //ID の入力
 Close;
 Unprepare;
 SQL.Clear;
 SQL.Add('SELECT ID,NAME,FURIGANA,PHONE,ADDRESS FROM KANJAMEIBO
WHERE ID=:ID');
 AtaiOfID:=StrToInt(EditKensakuNaiyou.Text);
 ParamByName('ID').AsInteger:=AtaiOfID;
 Prepare;
 Open;
 end;

 if   PhoneRadioButton.Checked=True then
 With QKensaku  Do begin                                //        電話番号
 Close;
 Unprepare;
 SQL.Clear;
 SQL.Add('SELECT  ID,NAME,FURIGANA,PHONE  FROM  KANJAMEIBO  WHERE
PHONE=:PHONE');
 AtaiOfPhone:=EditKensakuNaiyou.Text;
 ParamByName('PHONE').AsString:=AtaiOfPhone;
 Prepare;
 Open;
 end;

 if   NameRadioButton.Checked=True then
 With QKensaku  Do begin                                    //名前
 Close;
 Unprepare;
 SQL.Clear;
 SQL.Add('SELECT  ID,NAME,FURIGANA,PHONE  FROM  KANJAMEIBO  WHERE
NAME LIKE :NAME');
```

```
 AtaiOfName:='%'+EditKensakuNaiyou.Text+'%';  //  名前の一部でも可
 ParamByName('NAME').AsString:=AtaiOfName;
 Prepare;
 Open;
 end;

 if   FuriganaRadioButton.Checked=True then
 With QKensaku  Do begin                                    //ﾌﾘｶﾞﾅ名前
 Close;
 Unprepare;
 SQL.Clear;
 SQL.Add('SELECT  ID,NAME,FURIGANA,PHONE  FROM  KANJAMEIBO  WHERE
FURIGANA LIKE :FURIGANA');
 AtaiOfFurigana:='%'+EditKensakuNaiyou.Text+'%';   //ﾌﾘｶﾞﾅの一部でも可
 ParamByName('FURIGANA').AsString:=AtaiOfFurigana;
 Prepare;
 Open;
 end;

  if   JuushoRadioButton.Checked=True then
 With QKensaku  Do begin                                    //住所
 Close;
 Unprepare;
 SQL.Clear;
 SQL.Add('SELECT ID,NAME,FURIGANA,PHONE,ADDRESS FROM KANJAMEIBO
WHERE ADDRESS LIKE :ADDRESS');
 AtaiOfJuusho:='%'+EditKensakuNaiyou.Text+'%';   //住所の一部でも可
 ParamByName('ADDRESS').AsString:=AtaiOfJuusho;
 Prepare;
 Open;
 end;
```

解説

＊変数

```
Var AtaiOfID:Integer;
AtaiOfPhone:String;
```

```
AtaiOfName:String;
AtaiOfFurigana:String;
AtaiOfJuusho:String;
```
最初にパラメータ値を変数として宣言しています。変数は必ずデータ型を記述することが必要です。

（ID で検索する部分のコード）

```
if   IDRadioButton.Checked=True then
 With QKensaku  Do begin                          //ID の入力
 Close;
 Unprepare;
 SQL.Clear;
 SQL.Add('SELECT ID,NAME,FURIGANA,PHONE,ADDRESS FROM KANJAMEIBO
WHERE ID=:ID');
 AtaiOfID:=StrToInt(EditKensakuNaiyou.Text);
 ParamByName('ID').AsInteger:=AtaiOfID;
 Prepare;
 Open;
 end;
```

＊パラメータークエリー

次に ID **ラジオボタンがクリックされたときに、**ID **をもとに問いあわせをするために** ID **をパラメータとして** SQL **文を作成します。**

Qkensaku **はフォーム上に** SQL **コンポーネントとして配置していますので改めて変数宣言をする必要はありませんが、一時クエリーを作成する場合は** TSQL **として宣言します。**

はじめに SQL **文を削除し、パラメータとしての** ID **は**:ID **と表現します。**

パラメーターとしての ID **そのものに値を与えるためには**

```
ParamByName('ID').AsInteger:=AtaiOfID;
```

とします。ParamByName('ID').AsInteger **は** TSQL **の（この場合は** TSQL **のインスタンスとしての** Qkensaku **の）プロパテイです。**

パラメータは SQL **を開く前に与える必要があります。**Prepare **メソッドは記載がなければ自動的に呼び出されますので必須ではありません。**

＊型変換

AtaiOfID:=StrToInt(EditKensakuNaiyou.Text);**とあるのは** ID **が整数型であるのでテキストから整数型へ型変換を行う必要があるからです。**

ID を入力して問い合わせの結果を表示した場合は下図のようになります。