第12回 RAD Studio勉強会@Osaka用

RADプログラマーの為の データベースの構築方法

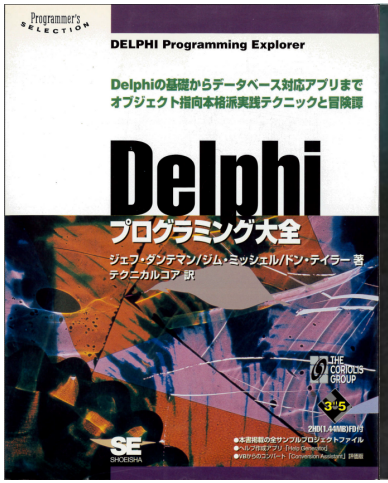
いつもお酒だけを楽しみにこの飲み会に参加しています。

RADプログラマー　板谷　吉洋

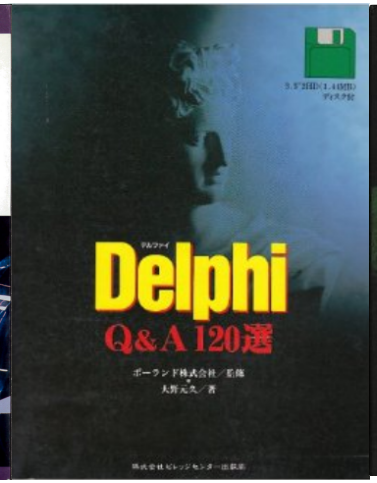
RADとはp5参照のこと

参考）お世話になった本10冊　もちろん他にもありますが…

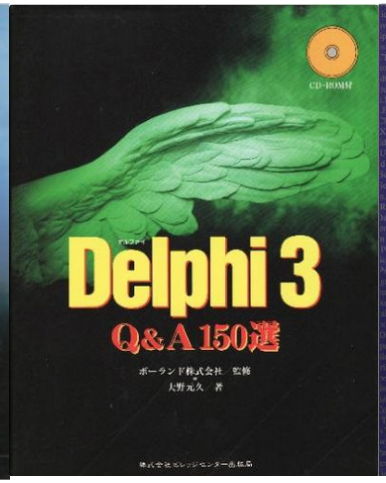
僕のDelphi入門書
物語風の説明がお気に！



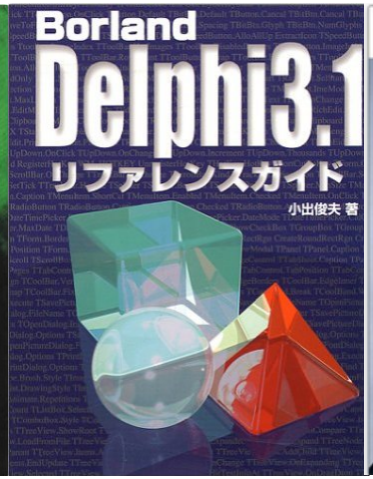
Basic、C言語からの移行時



WIndowsAPIの使い方

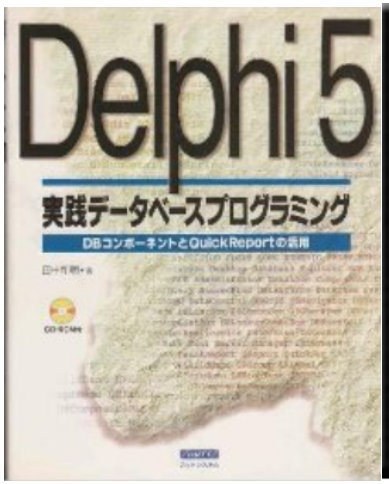


今も愛用

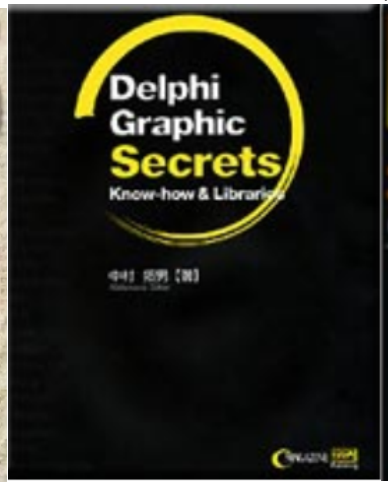


この本でCGIを勉強…

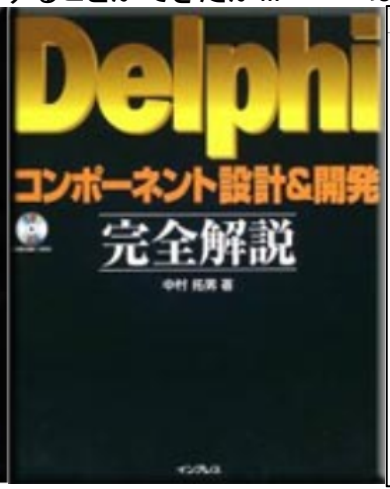
データベースはこれ？



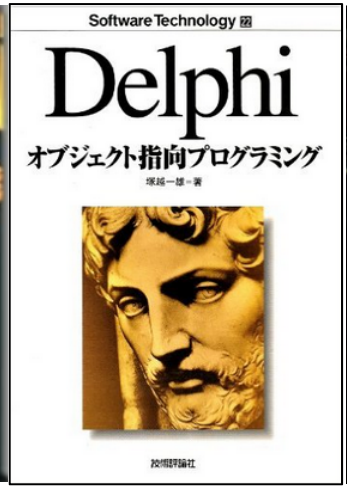
Graphicsの勉強用



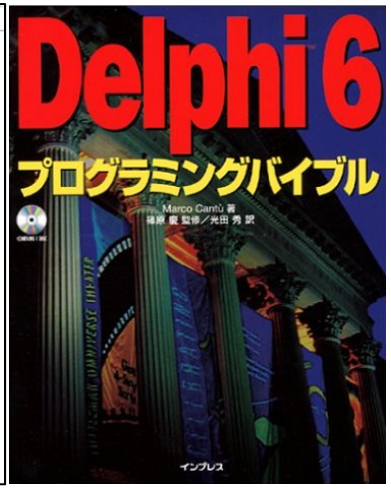
これでコンポーネント設計することができたが…



オブジェクト指向の考え方はこれから…



Marco Cantùの本だった

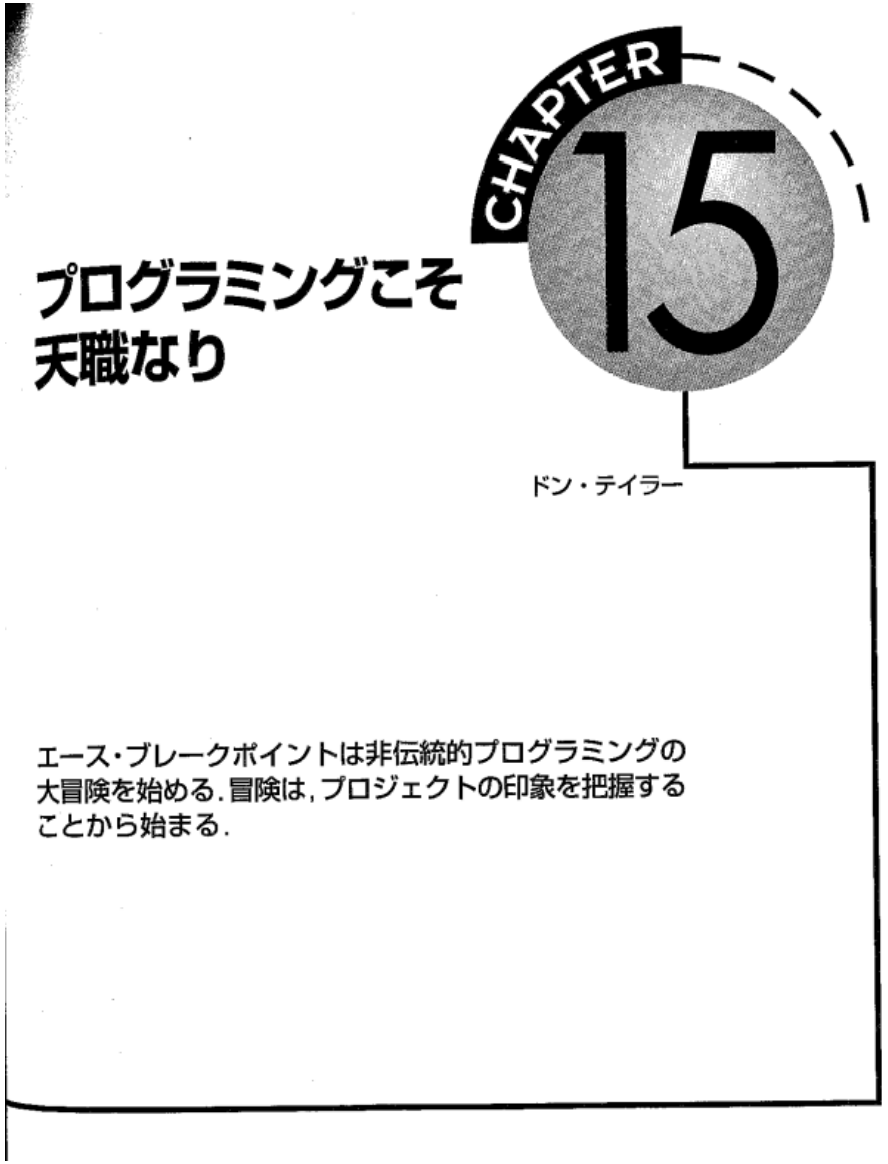


僕のDelphi入門書　Delphiプログラミング大全から

目　次

僕のDelphi入門書　Delphiプログラミング大全から

僕のDelphi入門書　Delphiプログラミング大全から



僕のDelphi入門書　Delphiプログラミング大全から

これまで，Delphiの強力な機能を見てきて，簡単なWindowsアプリケーションならばコードを1行も書かずに作成できることがわかりました．そして，オブジェクトPascalによるプログラミングの基礎を学びました．また，Delphiアプリケーションにコードをどのように追加すればその動作を完全にコントロールできるかもわかりました．

では，いよいよ探検に向けて出発です．

読者の中には，帳票アプリケーションを作るときの説明のような，細かくステップバイステップで進む，しかもそのスピードがものすごく遅い白習方式の説明を想像している方もあるでしょう．これで，すぐ思い浮かぶのは「たいくつ」という言葉です．もしあなたがそのような方法をお望みならば，あらかじめお詫びしておかなくてはなりません．本書はまったく違います．

私の最初のエンジニアリングの教授は，Ronという先生でした．その先生に徹底的にたたきこまれたことがあります．それは，大切なのは正解を得ることではない，「問題の解き方を知る」ことだ．設計においては，まさにこれが真実であり，「正しい」答えは広範囲に存在するのだ，ということです．

読者の皆さんとこれからひとつの「体験」を共有することになるのですが，他のチュートリアル(自習書)とは違います．またDelphiの使用方法も従来の開発ツールによるプログラミングと違います．チュートリアルと呼ぶことはできません．「冒険」そのものだからです．しかもこの冒険では，人が関わる度合を読者が自分で決めることができます．

まず，Delphiを初めて使うプログラミング・コンサルタントの架空のストーリーを読んでいただきましょう．話を通じて，そのコンサルタントの内面の心の動きに触れてください．彼の行動の一部始終，すなわち問題の定義から，アプリケーションとそれをサポートするデータベースの設計およびインプリメンテーションに至るまでを，逐一追っていきます．ストーリーをさらっと読めば，Delphi特有の問題の解決方法もいくつかはわかります．もっとのめりこんでいろいろ学びたいタイプの読者ならば，ストーリーに説明されているアプリケーションを，主人公と一緒に作っていってください．ひどく不精な人ならば，カウチポテトよろしく，このいくらかデキの悪いフィクションを読んでみてください．選択は自由です．

で，そのストーリーですが，フィクションではありますが，2つだけ本物があります．

ひとつは，クライアントは実在の会社です(「太平洋岸コミュニティにさんぜんと輝く星，Boxlight」という記事を見てください)．話の中に登場してくるBoxlight製品の情報と仕様は，実物の製品ラインのサブセットであり，本書が印刷される時点でその情報は正確です．ここに，Boxlight Corporationのご協力と，そして勇気に感謝いたします．「現実世界」らしい例を作ることができました．

2つ目は，パウルスボ(Poulsbo)は実在の町です．ワシントン州シアトルの西約15マイル(直線距離)に位置します．この町は，もともとはタフなノルウェー移民の起こした漁村でした．現在は観光の町

であるといってよいでしょう．マリーナや，古風な店やレストランがひしめく曲がりくねった街並みが，その特長です(ま，正直なところ，今はにぎやかになりすぎてあまり歩き回る気にはなりません．それによく雨のふるところです)．

この2つを除いて，ストーリーの中味は現実世界と何のつながりもありません．Boxlight社の機密に抵触するような情報，例えばベンダー，顧客，価格データなどは架空のものです．同様に、登場する地名や場所名もほぼフィクションです．登場人物も，実在または過去に実在した人物といっさい無関係です．

主人公の紹介

以上を背景にドラマは展開します．主人公はエース・ブレークポイント，精力的な，まさに小説に登場するような私立捜査官から，一転してプログラマーの道に進んだ男です．

ハックンサックで生まれ育ったエース少年はP.I.(Private Investigator，私立捜査官)になるのが夢だった．あの古き1940年代の映画に登場する，フィル・マーロウ，サム・スペード，エラリー・クイーンといった主人公のような私立捜査官である．しかし後年，きつい，ハードな捜査術とスマートな対話法を学んでからというもの，現代には1940年代のP.Iは少しも需要のない職業であるとエースは気付く．

エースはめげずに，転職を決意する．選んだのは，「一生」需要のありそうな職業，Windowsプログラマであった．とはいっても，目指したのは，単なる現代のプロフェッショナルよりももっと重要な，現代におけるひとかどの「人物」になることであった．90年代の人物になるために，タフな，いかにもガチガチな物腰を捨てようと思った．代わりに，気配り，繊細さ，オープンな性格，そして周囲との付き合いの良さを身につけようと思った．

パウルスボに移り住み，スクウェイミッシュ・オブ・アートの夜間のコンピュータ・プログラミング講座に通った．卒業式で，プログラミング適合性のアソシエーツ・オブ・アーツを授与される．そして，オフィスを借りて，開業することになった．

たいていの主人公同様，エースも小さな欠点をいくつか持っている．正式な教育を受けたけれども，最先端については若干の弱さがあること．精神力の強い私立捜査官になるために特訓していたころからのたくさんの荷物を，依然として引きずっていること．神経の細やかさが、安物のスーツ同様、なかなか離れてくれないこと．まちがいもしばしばやらかしてしまう．しかし彼には，粘り強さがある．それが彼の愛すべき長所だ．問題に直面したとき，巧みに身をかわしながら，追いつめ，や

僕のDelphi入門書　Delphiプログラミング大全から

がて解決する．彼がDelphiに最初出会ったときもそうであった．エースはまず覚悟を決める．

「ようし，このDelphiというやつをできるようになれば，人が何をやってのけるのか想像できるぞ」

と，このようにエースは納得します．エースの事件簿の中から次のストーリーを読んでいきますが，読者はエースと共に勇気を奮い起こしてください．Delphiを使った完全なアプリケーションのビジョンを描き，そして開発するという体験をしていきます．エースが開発するアプリケーションのサンプルを，本書の付録ディスクに納めてあります．

いままで本書をお読みになっておわかりのように，ジェフやジム，そして私は，何か新しいことに取り組むときに，何も退屈な道を選ぶ必要はないと考えています．ややもするとそういう状態に陥りやすいのですが，断じてそうであってはなりません．絶対に楽しくなくてはなりません．

退屈の罠にはまってはなりません．だから，ポップコーンでもつまみながら，部屋の照明をすこし落として，スクリーンにいすを近づけて，冒険への準備してください．冒険は題して，

「不誠実なデモ事件」
エース・ブレークポイントの事件簿より

太平洋岸コミュニティにさんぜんと輝く星，Boxlight

ハーブ＆スローン・マイヤー夫婦は，自宅でコンピュータ・プロジェクション・パネルの販売の仕事を始めたとき，まさかこれほどの大成長ビジネスになるとは少しも想像していませんでした．ところが10年もたたないうちに，Boxlight Corporationは，自宅を本社としていた事業から，プロジェクションディスプレイ産業で押しも押されぬトップクラスの会社へと発展します．

1985年，液晶ディスプレイ(LCD)パネルは市場に登場したばかりでした．マイヤーズ夫妻は，経営で幅広いバックグラウンドを持っていたので，コンピュータベースのプレゼンテーションの利点にいち早く気が付きます．

「私たちは，チャンスの窓をのぞきこみ，それを利用したのです」

とハーブは語っています．

第3弾目のLCDプロジェクションパネルを世に出したところで，夫妻はカリフォルニア州のマリン・カウンティの自宅オフィスから，ワシントン州バウルスボに移転します．1989年の11月でした．新しいオフィスはリバティ湾を見渡す波止場にあります．このオフィスになってから5年間で，スタッフは40人以上に増え，1994年の売上は1千万ドルの大台に乗ります．Boxlightの製品ラインはプロジェクション装置だけで50モデル以上．顧客ベースはフォーチュン500社にのる会社や，有名大学，官庁機関にまで広がっています．例えば，ディズニー，AT&T，IBM，フォード，マイクロソフト，そしてハーバード大学やアメリカ大統領にまで売られています．

1994年Boxlight社は，『Inc. Magazine』誌の急成長の独立企業500社に名を連ねます．5年間の成長率はなんと642パーセント．同年，『Washington CEO』誌が，米国で最も成長率の速いハイテク企業として同社を取り上げました．

成功の秘訣はどこにあったのでしょうか？　マイヤー夫妻は，テクノロジーに遅れをとらずにきたことの他に，際だったサービスで評判になった点をあげています．

「私たちは有能な，深い知識のある人材を雇いました．そして，顧客にサービスするためなら何をしてもいい，という権限を彼らに与えたのです．部品交換で徹夜になったり，その他のユニークなサービス中心のやり方で顧客にサービスすることもあるのです」(スローン)

Boxlight社はまた，受注後24時間で出荷することを保証することでも有名です．

マイヤー夫妻の願望は，会社を常に最高の状態にしておくことです．

「このようなハイテク市場では，実力を発揮できるか，脱落するかのどちらかです．中間は有りえません」(ハーブ)

こういう姿勢である限り，Boxlightの未来は非常に輝かしいものとなるでしょう．

私は『Delphi Windowsプログラミング』を閉じ，机の書類の山の上に放り投げた．パートIでは，Windowsプログラミングについてかなり勉強になった．パートIIを読み進めるうちに，すべてがわかりきったように思え始めた．しかし，自分がそこまで学んだことを，ある重大なテストで試すことになろうとは，このとき知るよしもなかった．

単調なスタッカートのように雨滴が窓を打っている．これがもたらす一種の催眠効果と，1時間前から飲み始めてすでに2杯目のコロンビア・シュープリームとが，ほどよく中和されてきていた．目は中空を見つめ，頭はダイアログのボタンを使い方じっと思考を集中させていた．と，その時，電話が鳴っているのに気が付いた．

机の上の立て置き電話を左手でつかんで，パッっと上に放り上げ，右手でスパッと受け止め，受話器を口に近づけた．

「エース・ブレークポイントです．どういったご用件ですか？」

あれ？　まだ電話が鳴り続けているゾ．なんてこった．これは，めいっ子のロドニーが去年のクリスマスで私にくれた電話貯金箱じゃないか．貯金箱を机に落とし，書類をかき分けて電話に出た．

「エース・ブレークポイントです．どういったご用件でしょうか？」

参考）お世話になったサイト…　この場を借りて

- 公式サイト）やっぱりこれ！
- Embarcadero　日本語のフォーラム　:もしかしていまない？
- Embarcadero　チュートリアル

- 非公式サイト）教えてくれて有難う
- Delphi Library [Mr.XRAY]
- Googleってヒットした皆様

１）使ったデータベース

- FileMakerPro：カード型データベース　1990年頃～
- Excel：表型データベース　1990年頃～
- Oracle：リレーショナルデータベース　1995年頃
- （Access：Visual Basicで使うため）　1995年頃

- BDEとOracle:Delphi2～Delphi6　1996年頃～2001年
- MYDataBase:Delphi7～XE5　2002年～2013年
- FireDacとSQLite：DelphiXE5　2016年～

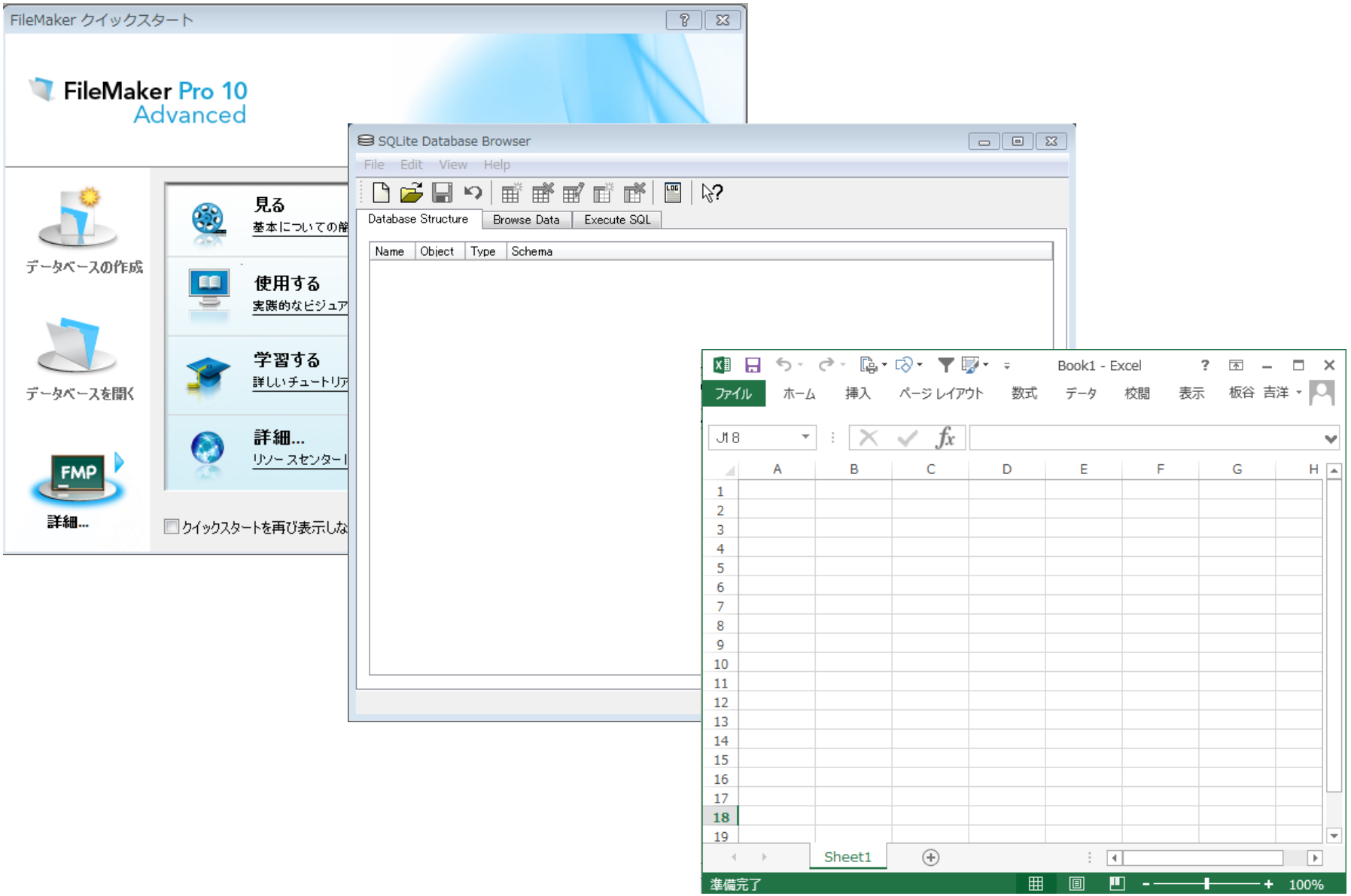
２）FileMakerPro：私の一番のお気に入り　いいところ

- 利点）
- GUI的にDBが作成出来る：いい加減なプログラマー向きアプリ
- 大概のことはここで作ることが出来ます。リレーショナルDBのテストモデルを作るには便利
- 強力なレポート機能：いちいちプログラムしなくてもすぐに出来てしまう。簡単なことは開発後ユーザーにお任せできる。プログラマー不要か？
- ２次利用も簡単（Excel並に…）

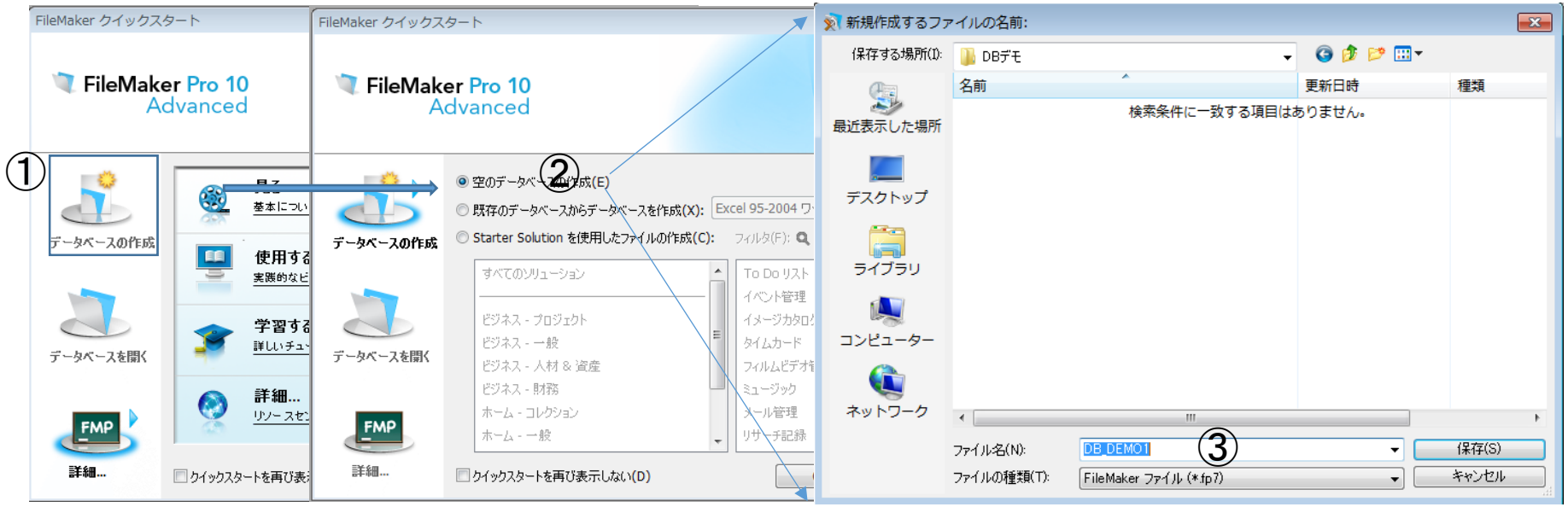
２）FileMakerPro：私の一番のお気に入り　だめだめなところ

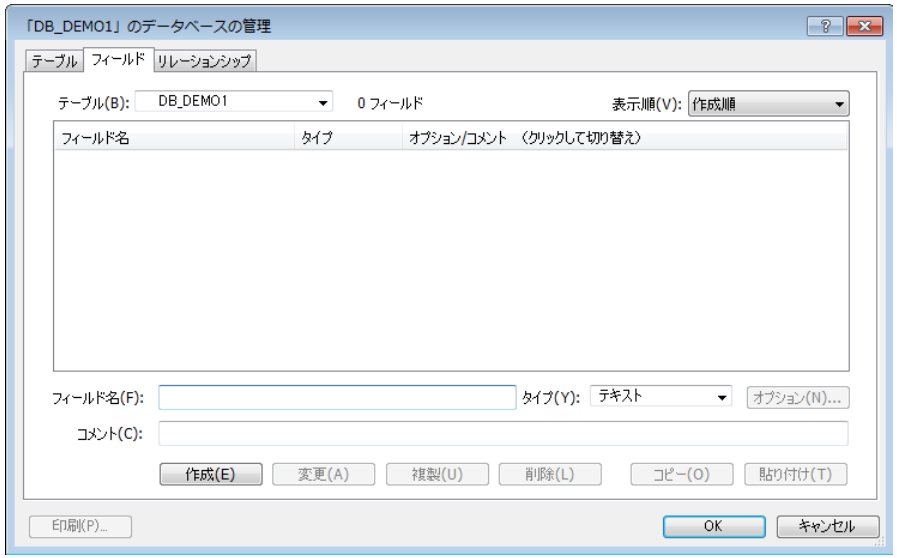
- 欠点）
- 大きなDBでは動きが遅くて使い物にならない。いまのCPU、メモリー、SSD、OSならかなり改善されているはず。
- 有料アプリ、１ユーザー３万程度
- プログラム言語に組み込めない。ファイル渡し
- 使っているユーザーが回りにきっといない。

3）かんたんデータベース作成



3）例えば FileMakerPro等のDBでのノーマル設計

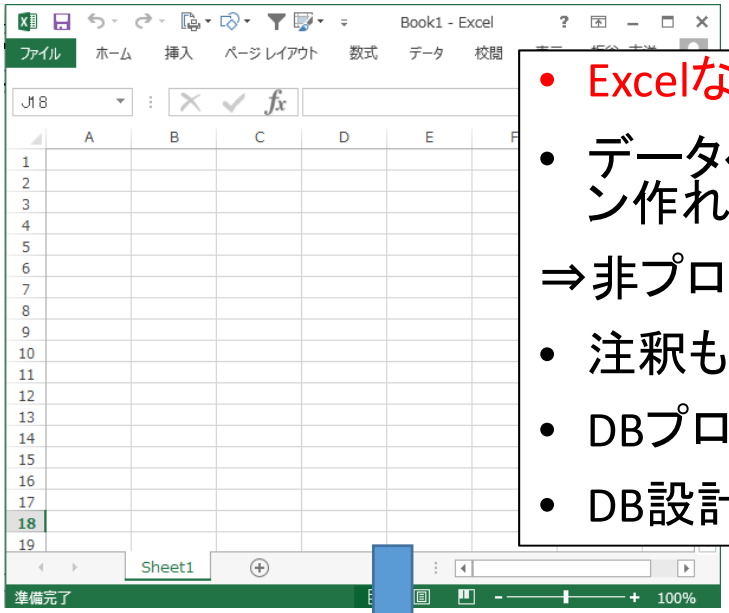




⑤ 大体いつも考えていなのでいきなり挫折！
全体を考えずに作りながら決めていくので。。。
典型的なスパゲッティープログラマーです

MYSQL,ACCESS,SQLite、Paradox…も面倒くささは私にとって基本同じ

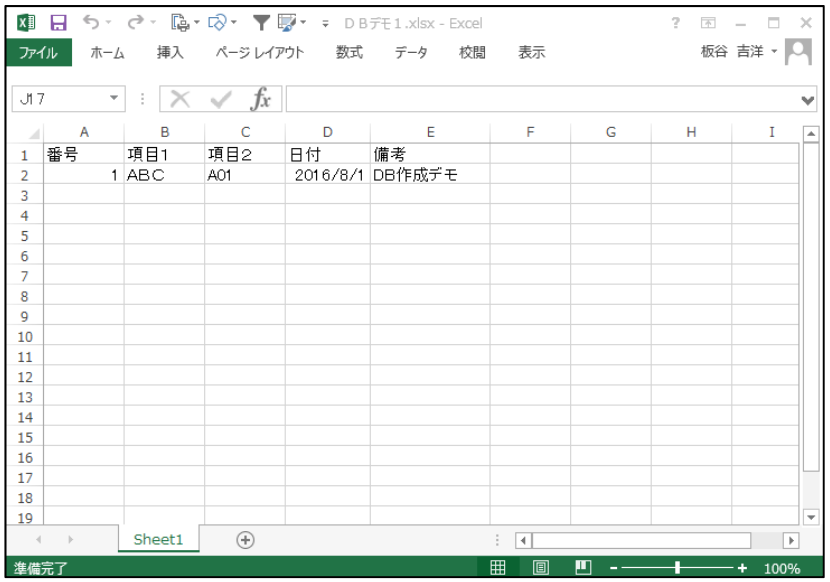
3）Excelを使おう！⇒　いちばん簡単に作成できるDB



- Excelなら誰でも使える。今の日本の会社員なら…
- データベースの構造が分からなくても目視でガンガン作れて、ガンガンテスト出来る。

⇒非プログラマーの方のでも設計に参加できる

- 注釈もがんがん書き込める
- DBプログラムを作れない人にでも扱える。
- DB設計中でもみんなで議論できる

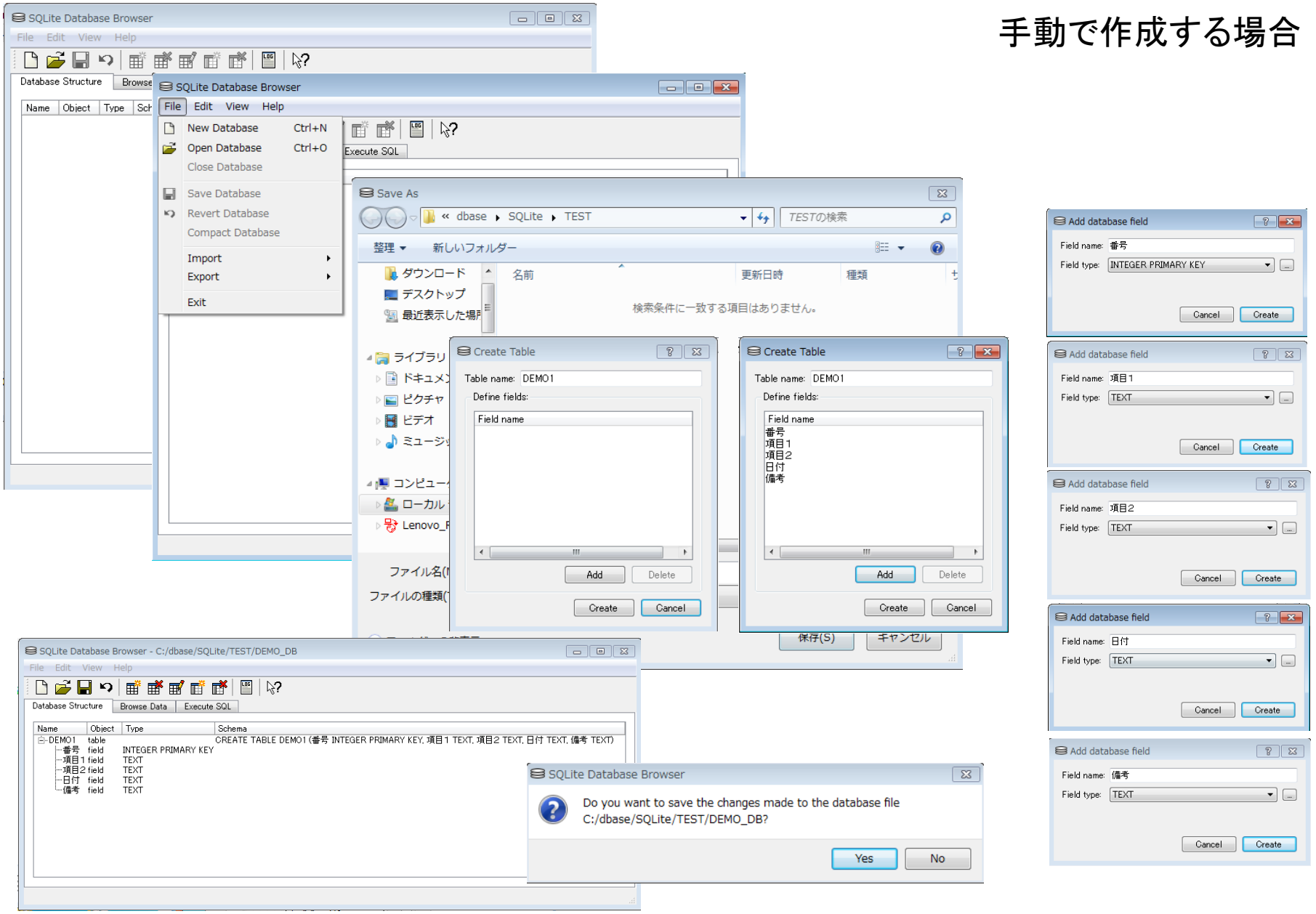


構造が分からなくて、分かっているところから、
①行目に項目名を決め、
②行目にサンプルデータを入れていけば良い
まずは、これだけです！

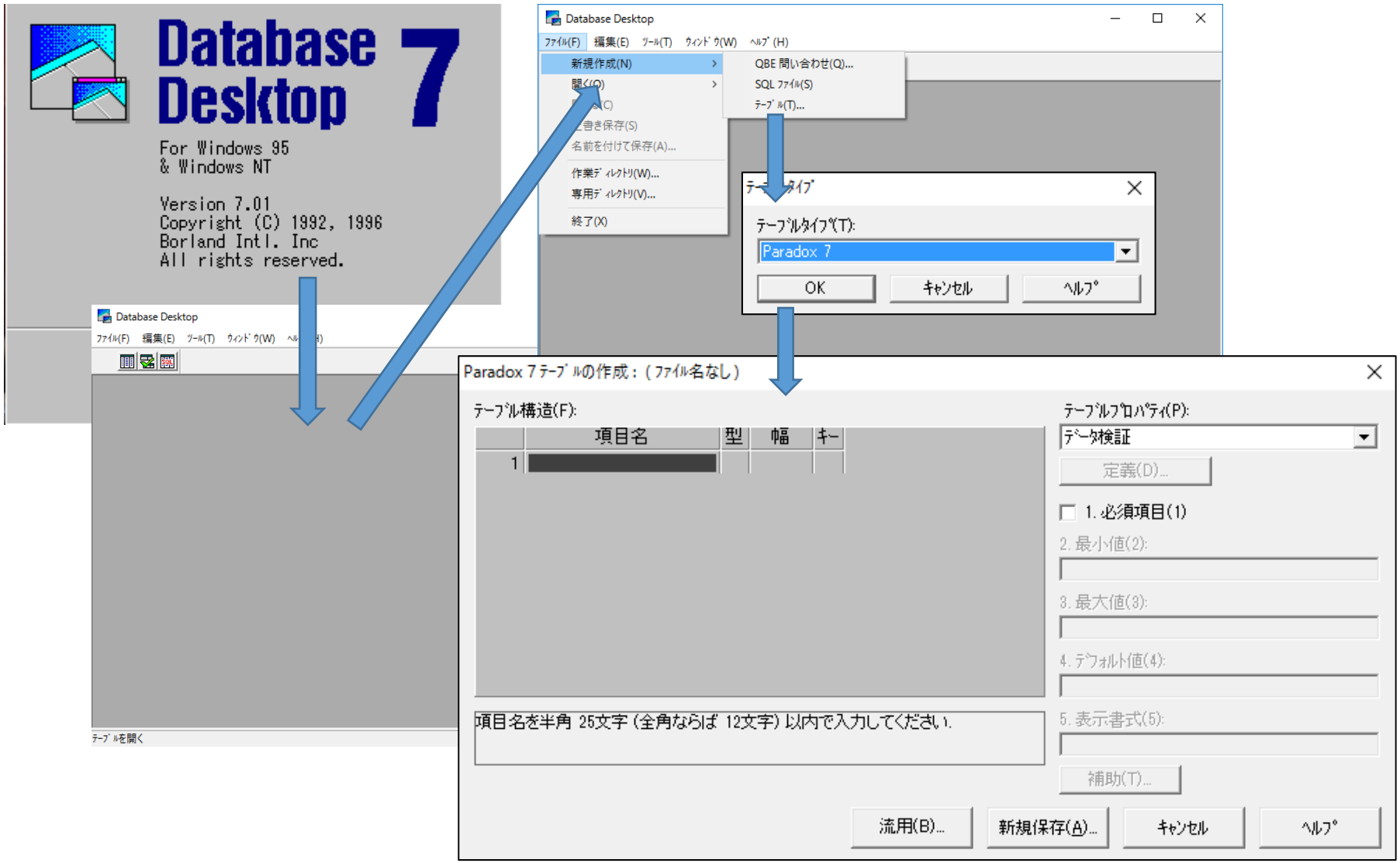
思いつく項目とそのサンプルデータが入力できたら、プログラムが完成した気になってしまいます。

3）SQLite：Embarcaderoさんの移行のおすすめのひとつだったので採用
項目名が決まったらDB Browser for SQLiteでDB（もちろんローカルで）作成

手動で作成する場合



3）DatabaseDesktop　＝　Paradox　懐かしい！
　以前は項目名か決まったらGUIツールでDB（もちろんローカルで）作成



３）余談：DatabaseDesktop　の困っていたこと

１）　TableかSQLか忘れましたが（多分SQL）、Open、Closeを繰り返していると、知らない内に、（きっと使い方がおかしかったのだと思っていますが…）

_QSQ1.DBと_QSQ1.MB

が出来てしまい、多くなると動かなくなる。

その為バッチ処理で下記の削除を実施して回避していました。

del *.db

del *.MB

２）　レコード数か容量がある程度大きくなると上手く動作しなくなっていた

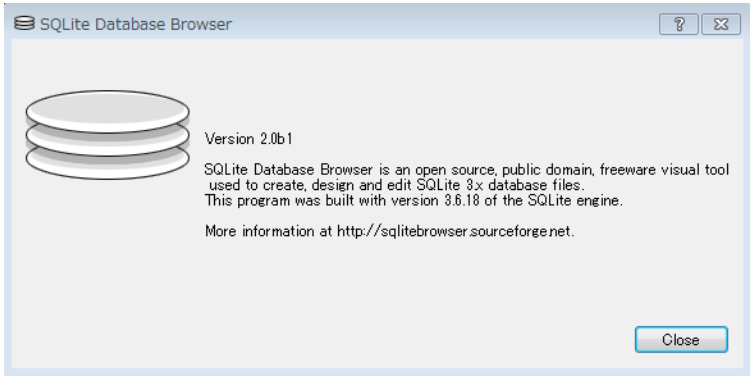
解決策は分からずに、StringGridへ移行していきました

3）余談：SQLiteでの困っていたこと

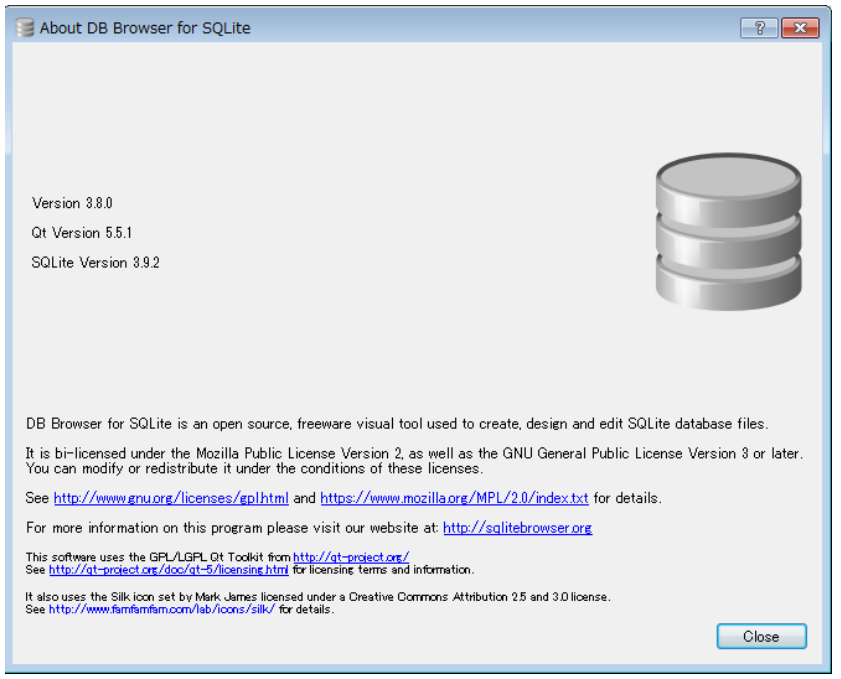
1）　レコード数か容量がある程度大きくなると上手く動作しない。レコードが表示できなくる。小規模はOK

2）　これを書き初めてDB Browser for SQLite 3.8.0のリリースに気が付きました

解決しているようです！



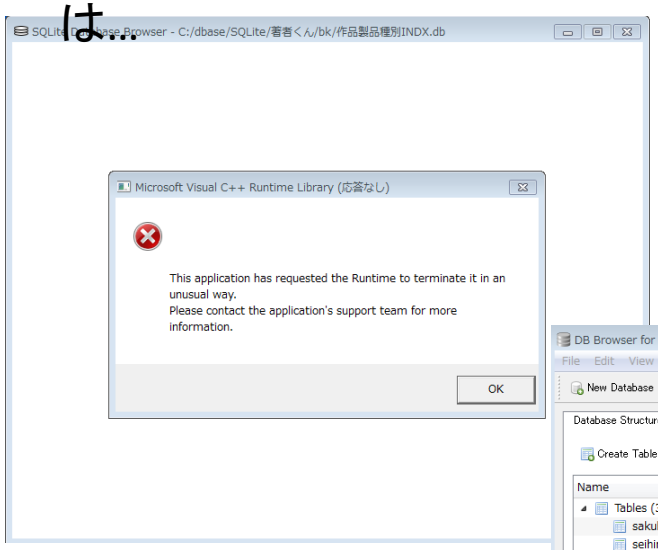
バージョンUPしましょう！



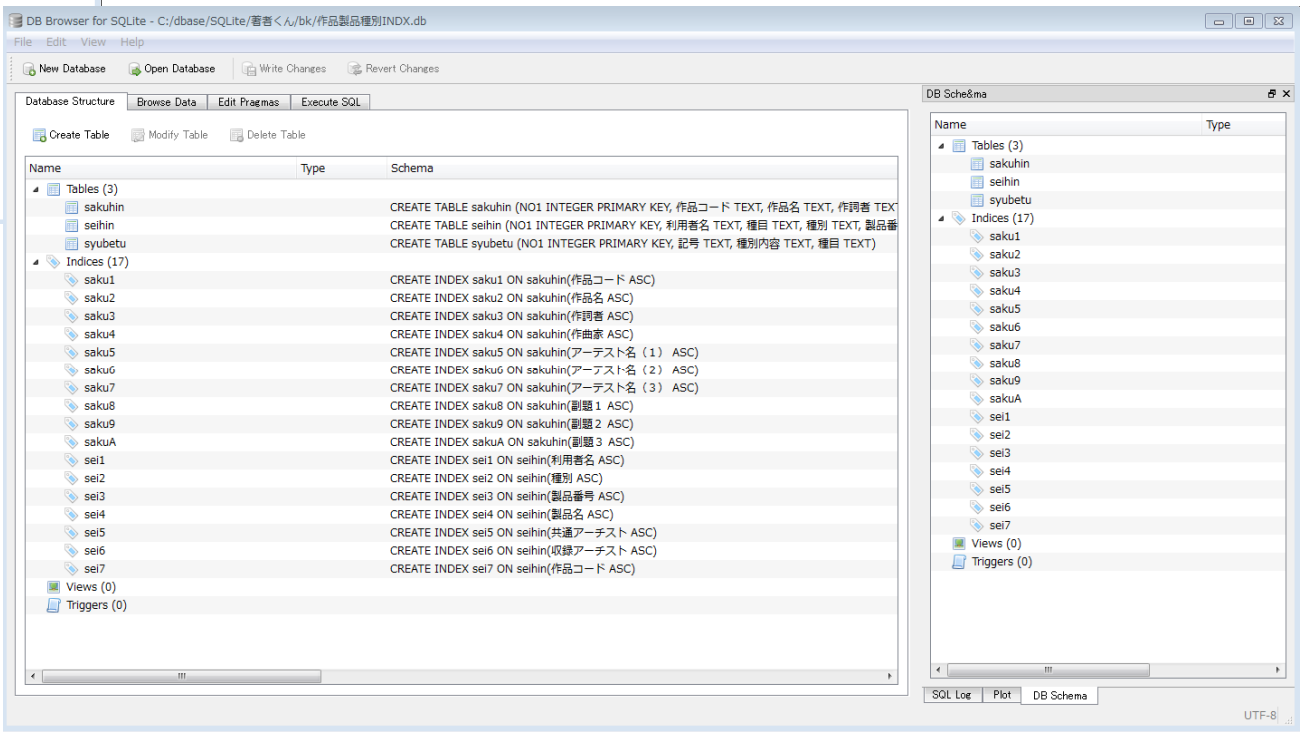
3）余談：SQLiteでの困っていたこと

作品製品種別INDX.db　2015/11/13 9:59　Data Base File　6,645,799 KB

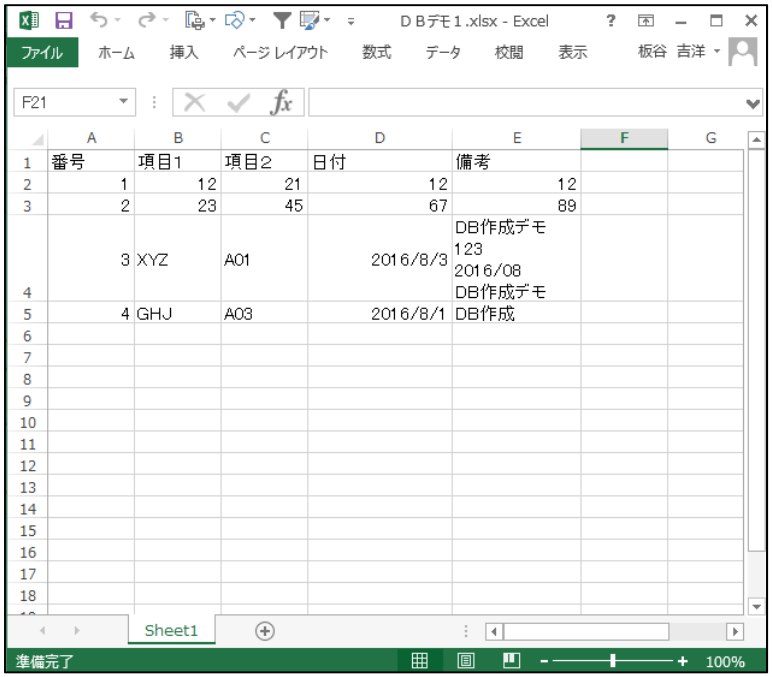
Ver2.0b1　：例えば6.645GBのサイズのファイルでは…



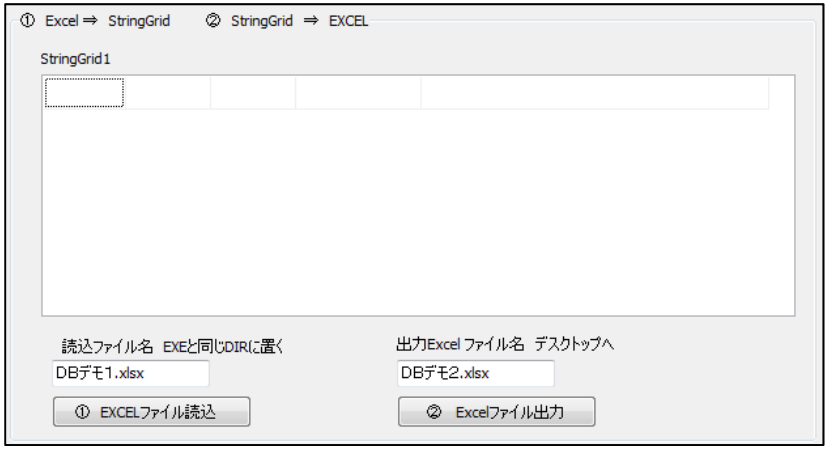
Ver3.8.0：巨大なデータでも動いている！感激！

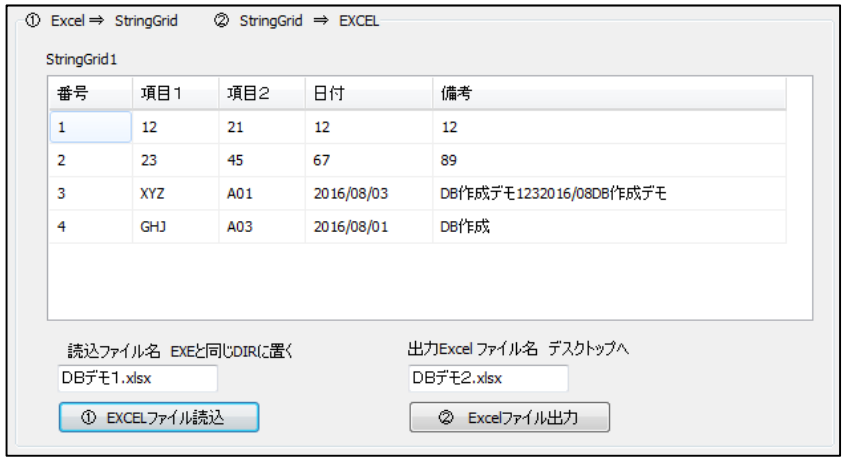


4）Excelファイル　⇒　StringGrid
ここからDelphiの世界でプログラミングがスタート



番号	項目1	項目2	日付	備考
1	12	21	12	12
2	23	45	67	89
3	XYZ	A01	2016/8/3	DB作成デモ 123 2016/08 DB作成デモ
4	GHJ	A03	2016/8/1	DB作成





1）Excelで作ったテーブルをStringGridへ
2）StringGridのデータをExcelへ

4）① Excelファイル ⇒ StringGrid

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Err,Filename,DIR:string;
  St:Integer;
begin
  GetDir(0,DIR);
  Filename:=DIR+'¥'+Edit1.Text;          //Excel File
  St:=1;                                 //Sheet 番号
  Excel_SG(StringGrid1,St,Filename,Err);  //Excel to StringGrid
  if length(Err) > 0 then Memo1.Text:=Err;
end;
```

読込条件

1）3000000行、10000列まで

きっと自動取得出来るはず…

⇒知っていたら教えて下さい。

今は1項目目が空白でbreak

2）1行目は項目名 （決め事です）

3）1列目はレコードの番号

4）Uses ComObj ：Excel用に追加

```
implementation
uses ComObj;

{$R *.dfm}

procedure Excel_SG(SGrid:TStringGrid ; St : Integer ; Filename,Err: string);
var
  i,j,x,y,fini_row,fini_col,iSt:integer;
  s,s1:string;
  Excel       : Variant;
  EApplication : Variant;
  WorkBook    : Variant;
  WorkSheet   : Variant;
begin
  Excel := CreateOleObject('Excel.Application');
  Excel.Visible:=false;
  WorkBook :=Excel.WorkBooks.Open(Filename,  true);
  WorkSheet := WorkBook.WorkSheets[St];
  //
  try  //WorkSheet1をアクティブにする。
    WorkSheet := WorkBook.WorkSheets[St];
    except
    on EOleException do
    begin
      Err:=' WorkBook.WorkSheets['+inttostr(st)+'] のシートがありません';
    end;
  end;
  y:=1;x:=1;
  for j:=  1 to 3000000 do    //最大3000000行  とりあえず
  begin
    s1:=WorkSheet.Cells[j,1]; //y  番号はユニークで空白なしがルール
    if length(s1) = 0 then
    begin
      fini_row:=j;   //空白までが有効
      break;
    end;
  end;
  for i:=  1 to 10000 do     //最大1000行  とりあえず
  begin
    s:=WorkSheet.Cells[1,i];  //x  項目名はユニークで空白なしがルール
    if length(s) = 0 then
    begin
      fini_col:=i;   //空白までが有効
      break;
    end;
  end;
  SGrid.ColCount:=fini_col-1;   SGrid.RowCount:=fini_row-1;
  for j := 1 to fini_row do begin    // y
    for i := 1 to fini_col do begin  // x
      s:= WorkSheet.Cells[j,i] ;
      SGrid.Cells[i-1,j-1]:=s;
    end;
  end;
  WorkBook.close;
  WorkBook:=unAssigned;
  WorkSheet:=unAssigned;
  Excel.Quit;
  Excel:=unAssigned;
end;
```

4）② StringGrid ⇒ Excelファイル

これがあると何かと便利

```
procedure TForm1.Button2Click(Sender: TObject);
 var
   Filename:string;
 begin
   Filename:=Edit2.Text;
   ExportToExcel(StringGrid1,Filename); //Excel 出力
 end;
```

やはりStringGridはそのままExcelファイルに出力したい。
非プログラム系の方とやり取りにはこれを配布！
（Excelの呪文）

```
var
  MsExcel,MsApplication,WBook,WSheet: Variant;
begin
 //Excel起動
  MsExcel := CreateOleObject('Excel.Application');
  MsApplication := MsExcel.Application;
  MsApplication.Visible := True;
  WBook  := MsApplication.WorkBooks.Add ;
  WSheet :=WBook.ActiveSheet;
```

コーディング

```
  pc_id:=GetUserNameS; //ログインユーザー名取得
  WBook.SaveAs('C:¥Users¥'+PC_ID+'¥Desktop¥'+Filename);
{後始末}
  WBook.close;
  WBook:=unAssigned;
  WSheet:=unAssigned;
  MsExcel.Quit;
  MsExcel:=unAssigned;
```

```
function GetUserNameS:string;
var
  NameBuff : array[0..255] of Char;
  NameSize : Cardinal;
begin
  NameSize := Length(NameBuff);
  if GetUserName(NameBuff,NameSize)
    then Result := NameBuff
    else Result := '';
end;
//
procedure ExportToExcel(SGrid:TStringGrid ;Filename: string); //Excel出力
var
  //
  MsExcel,MsApplication,WBook,WSheet: Variant;
  iCol,iCol1,iRow ,coun,coun1    : integer;
  //
  i,coun2:integer;
  s,s1,PC_id:string;
  ss:TMyArray2;
  //
begin
//Excel起動
  MsExcel := CreateOleObject('Excel.Application');
  MsApplication := MsExcel.Application;
  MsApplication.Visible := True;
  WBook  := MsApplication.WorkBooks.Add ;
  WSheet :=WBook.ActiveSheet;
  //Excelにデータ出力
  WSheet.Rows[1].Font.Bold:= 'True';  //タイトル行を太字にする
  coun:=SGrid.RowCount-1;
  coun1:= SGrid.ColCount - 1; //デスク担当で+1
  //
  for iRow:=0 to coun do begin
    for iCol:=0 to coun1 do begin
      WSheet.Cells[iRow+1,iCol+1].Value:=SGrid.Cells[iCol,iRow]; //番号
    end;
  end;
  //
  //
  pc_id:=GetUserNameS; //ログインユーザー名取得
  WBook.SaveAs('C:¥Users¥'+PC_ID+'¥Desktop¥'+Filename);
  //
  //
  {後始末}
  WBook.close;
  WBook:=unAssigned;
  WSheet:=unAssigned;
  MsExcel.Quit;
  MsExcel:=unAssigned;
end;
```

4）StringGridをMYデータベースにするための検索機能

①:検索文字を含んでいるレコード
②:検索文字を含んでいる項目と項目内の文字の位置
③:①の中での文字を表示（赤色等）
④:特定項目内での検索文字の表示

４）MYデータベースの利点/欠点

（利点）

- とにかく大量のレコード数でもOK

  実績で1.5GBは問題なく動く
- 検索がとにかく速い。SQLiteでindexを使用してみたが、まだまだこちらのほうが速かった。
- 自己設計なので、トラブル対策は簡単なのと、単純な為か、ほとんどトラブル知らず。。。

（欠点）

簡単に構築することが。。。

多重検索は作るのが。。。

4）DB Browser for SQLiteでDB作成

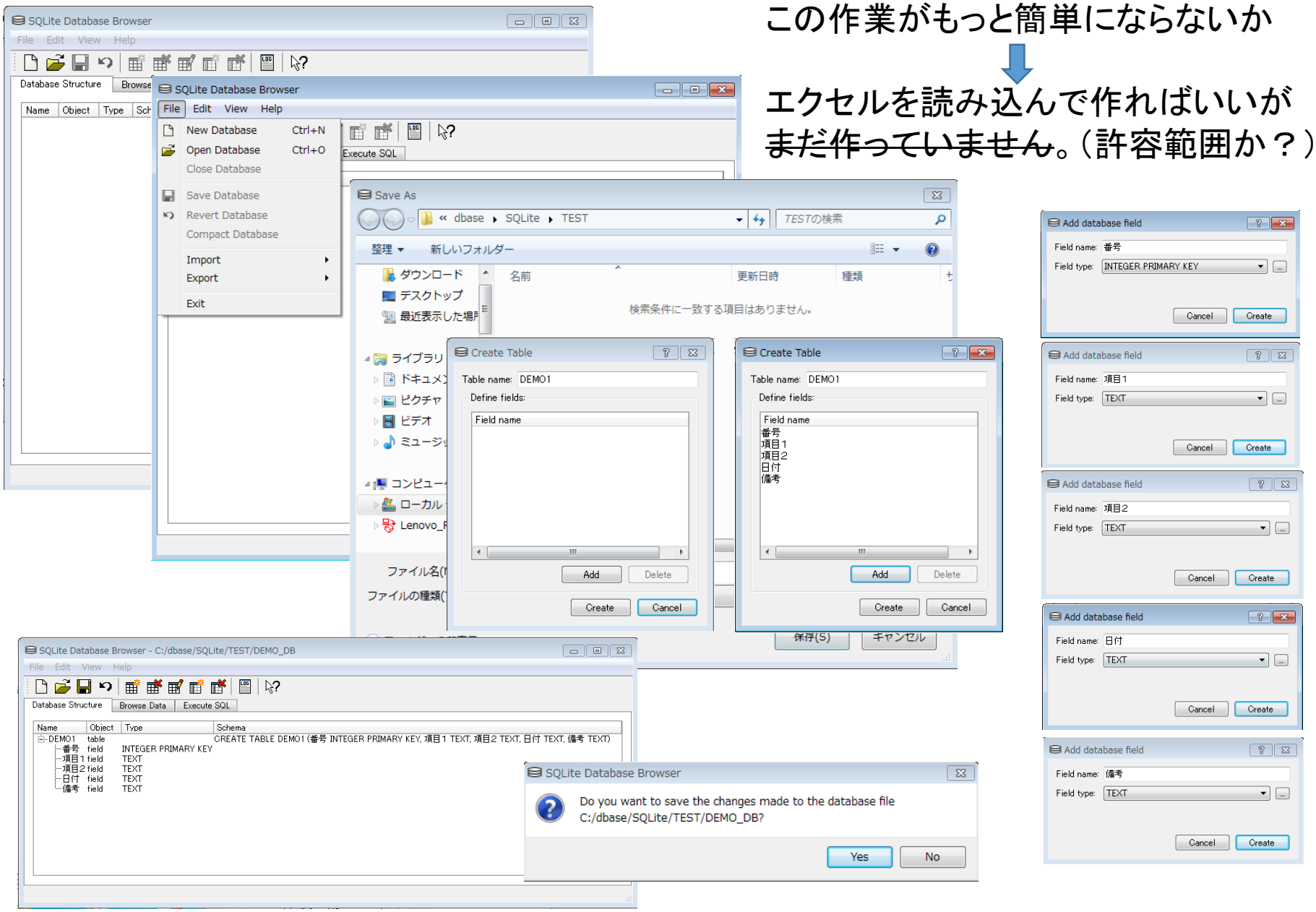
EXCELで項目名が決まったら作成

この作業がもっと簡単にならないか

↓

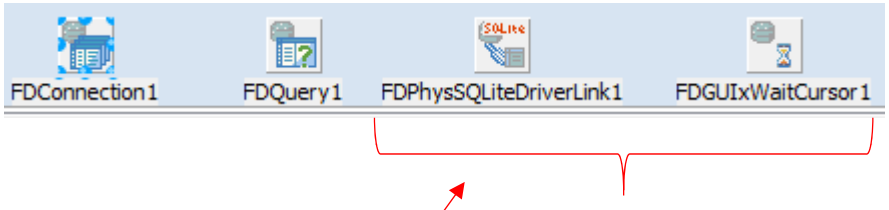
エクセルを読み込んで作ればいいが

~~まだ作っていません~~。（許容範囲か？）

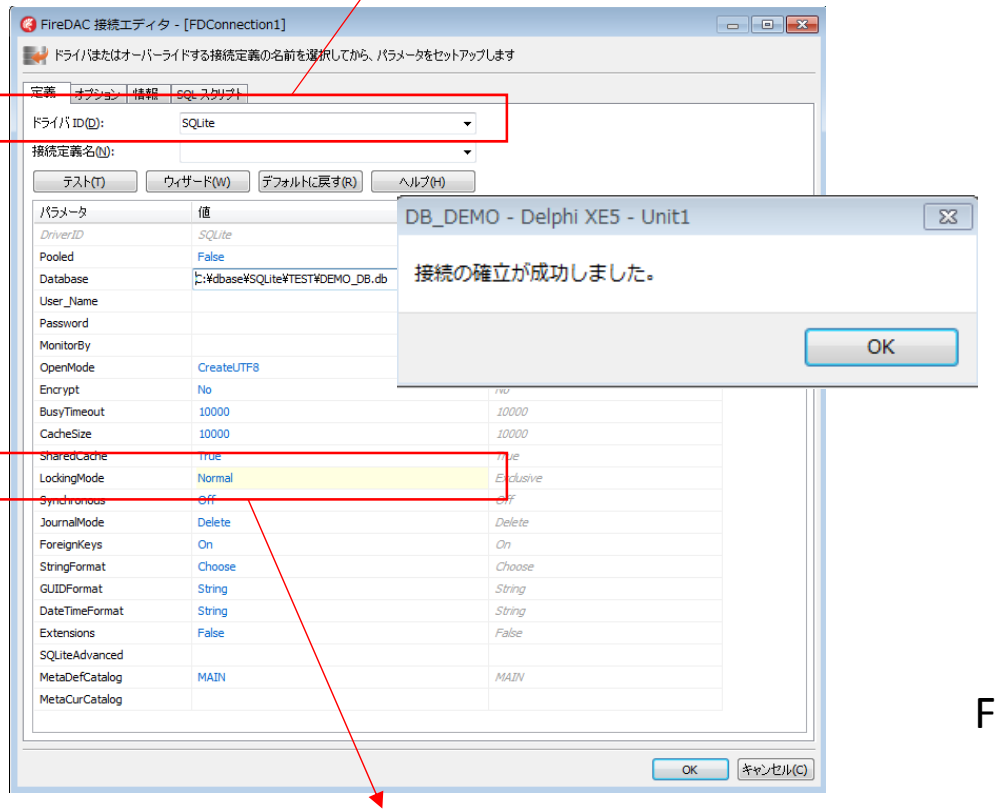


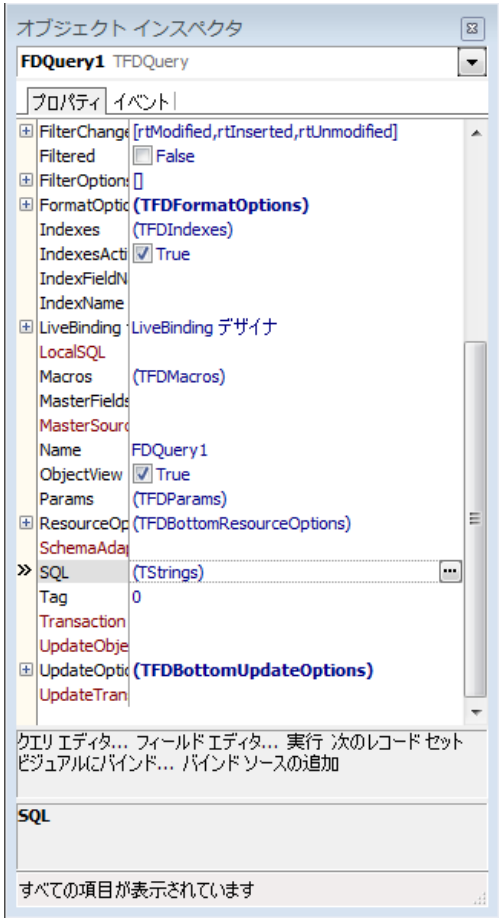
4）FireDac（SQLite）　まずは基本コンポーネントセット

儀式（お経、呪文の類）のようなもの１



くどい！：ここをセットしたら自動で…

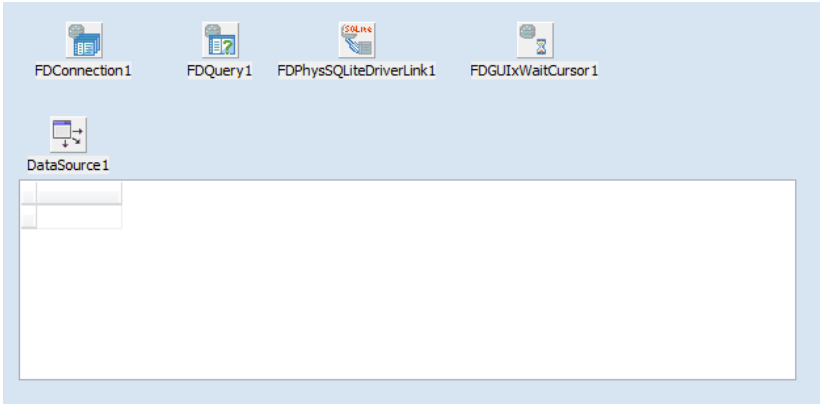




FDQuery1　SQL :select * from DEMO1

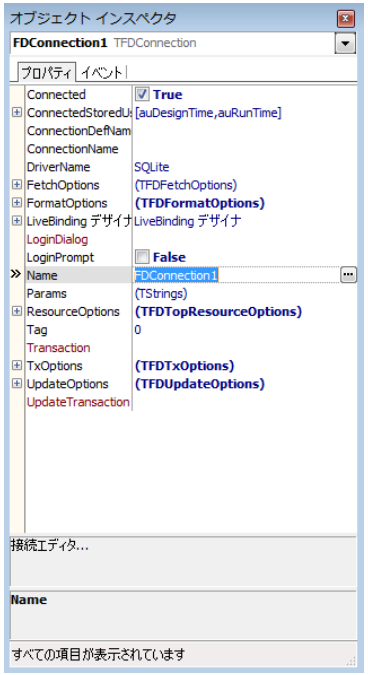
よく分かっていないが、Embarcaderoの紹介ビデオでNormalが選ばれていたので…

4）FireDac（SQLite）　儀式（お経、呪文の類）その2
事前に作ったDBをDataGridで表示させるまで

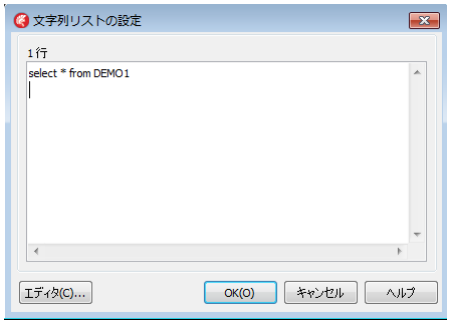


DataSource1
Dataset:FDQuery1

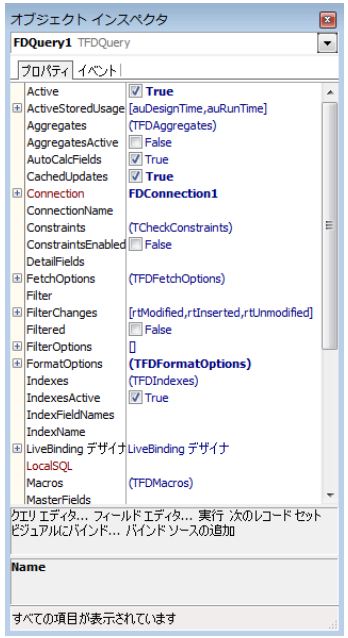
FDConnection1
Connected :True



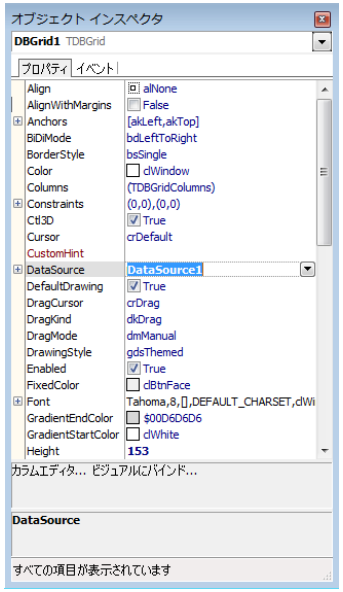
FDQuery1:
SQL :select * from DEMO1



FDQuery1:
Active:True

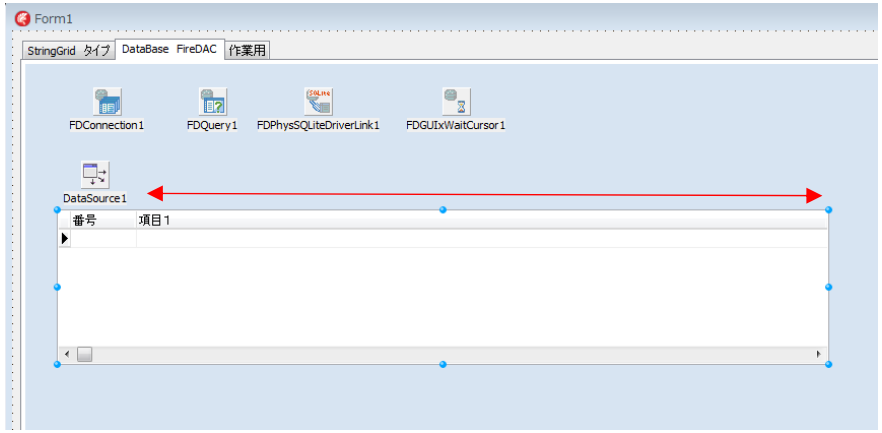


DBGrid1
Datasouce:DataSource1

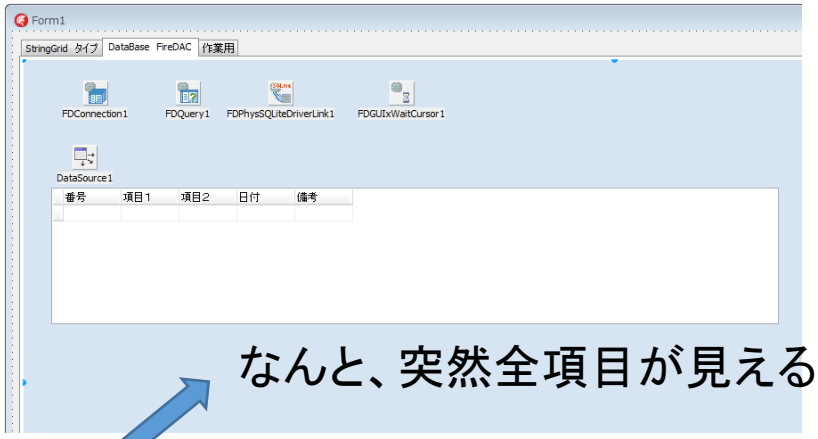


4）FireDac（SQLite）　儀式その3

だめだめ問題1：カラムの幅の調整がしにくい



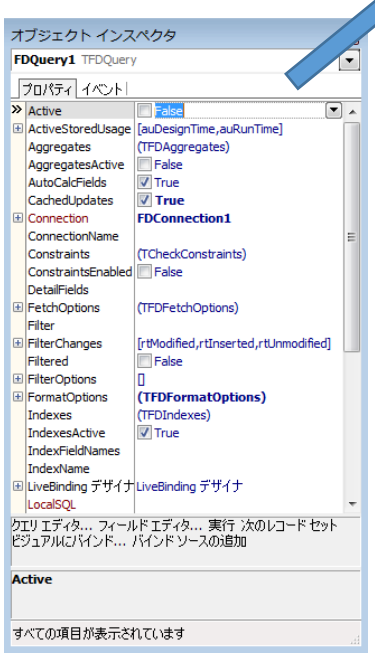
テキスト項目の幅がメチャ長い！
調整がしにくい

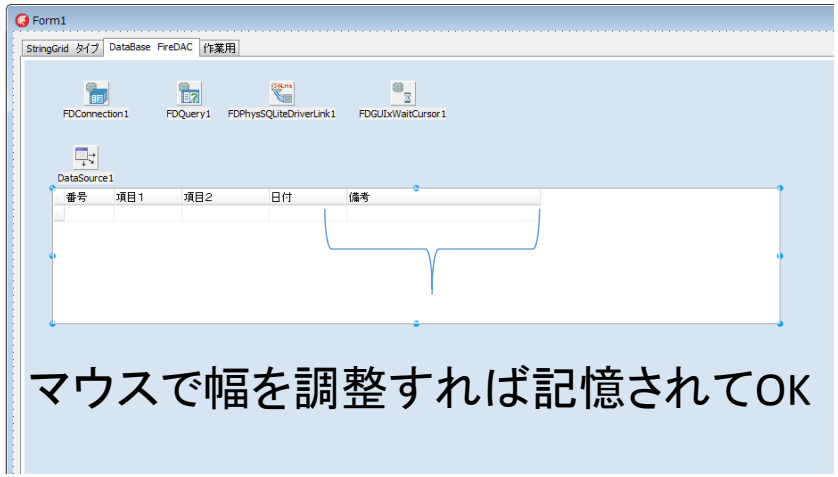


なんと、突然全項目が見える

FDQuery1:
Active:False

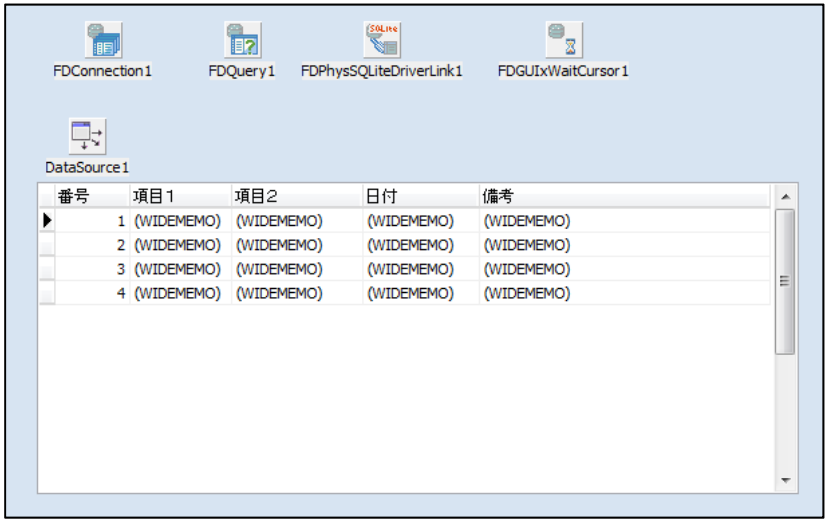
カラムエディターを利用





マウスで幅を調整すれば記憶されてOK

4）FireDac（SQLite）　DBをDataGridで表示させる
だめだめ問題2　⇒　解決編



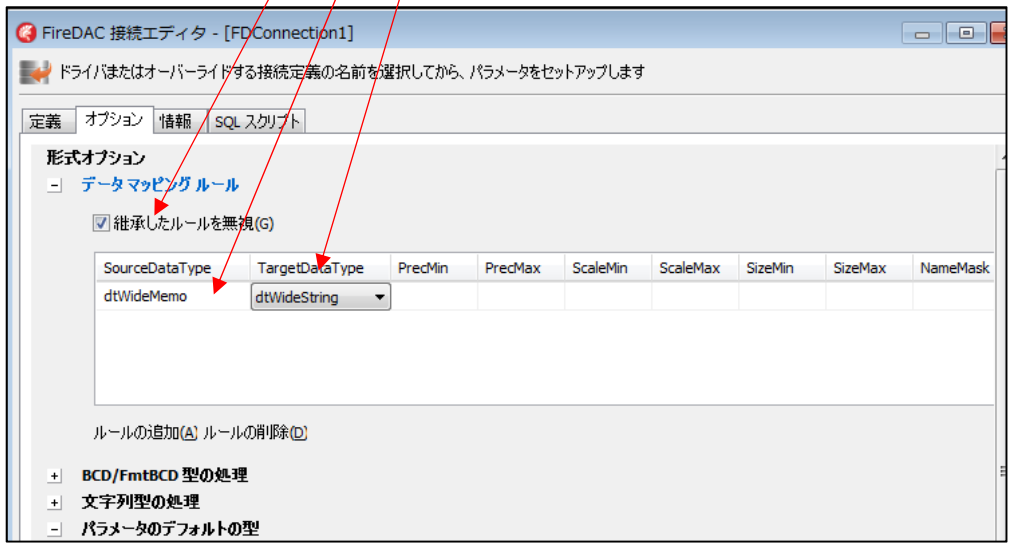
これを解決する方法だけど。。。
移行についての考え方が間違っていると思う。
FDDBGridがあるならいいけど
解決方法が提示されていないのは。。。

変更箇所はこの3つ
1）継承したルールを無視
2）dtWideMemo
3）dtWideString

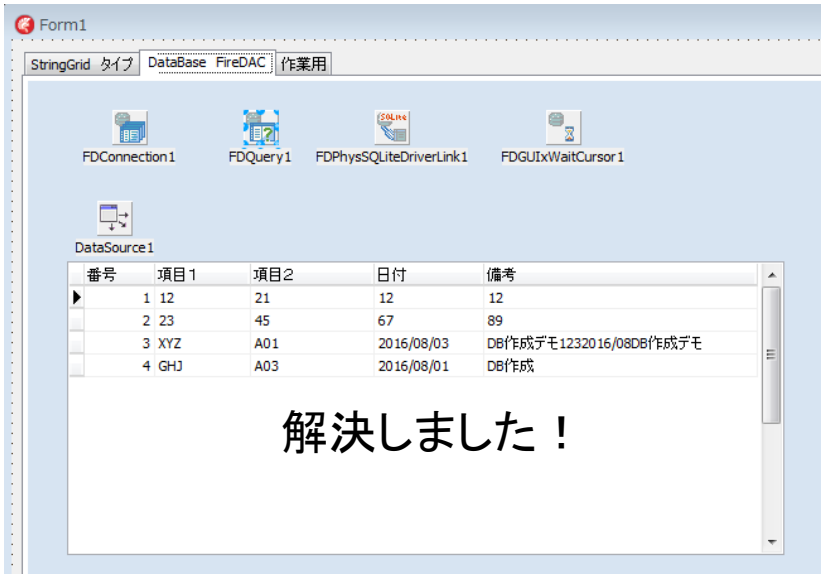
SQLiteでテキストは単に接続すると
（WIDEMEMO）と表示される

これを解決する方法はネットでいくつか
あるが、完全なものは無かった（後で説明あり）

第10回Embarcadero@大阪のメイン会議
の席（飲み会）にて愚痴ったところ
FDConnection1オプションにて
WIDEMEMOをWIDESTRINGに手動セットすれば
解決する方法を教授されました。



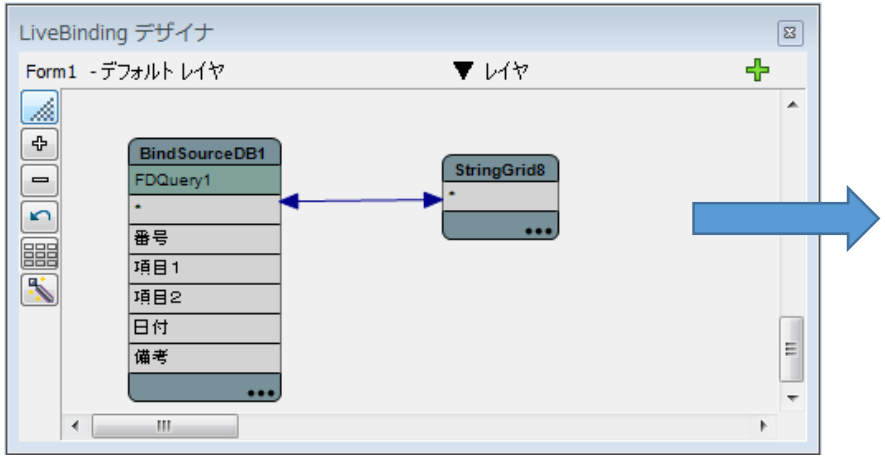
4）FireDac（SQLite）　DBをDataGridで表示させる
だめだめ問題2　　解決方法と以前のやりかた



これでようやくDatabaseDesktop
からFireDac＆SQLiteに置き換えられる
様になりました。

DBGridは（WIDEMEMO）表示のままだが

参考）以前の方法　ビジュアルバインドを使う

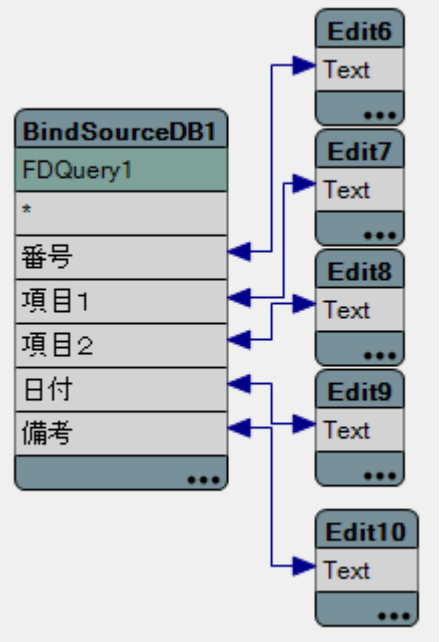


StringGridはノーマルな表示

4）FireDac（SQLite）　DBをDataGridで表示させる

ビジュアルバインド編　だめだめ問題3

- １）LiveBindingデザイナはコンポーネットが少ない時は見やすいが、多くなるとカオス！どこにあるのか分かりにくい。
- ２）StringGridへビジュアルバインドは行数が200まで。何故でしょうか？
- 　　200レコード以上だと表示がおかしくなる（データはOKだが。。。）
- ３）　２）改良版として、Editとビジュアルバインドすればokですが、StringGridに表示するには全レコードを先頭からEditを通して読まないと行けない　⇒　めんどくさいし、なによりスマートでない



Editでは変換されて内容が見える

4）FireDac（SQLite）ネットでの回避の仕方
DBをDataGridで表示させる　だめだめ問題４

Delphi：DBGridのデータが(WIDEMEMO)と表示される場合の対処法

組み込みデータベースのsqliteをADOで接続してselectしたテーブルをDBGridにバインドしてみたら、sqliteのTextフィールドの値が表示されないのでかなりハマった。WIDEMEMOって何？とか思っていろいろググってみたが解決方法はわからず。さんざん調べてみて解決方法はどうも2つあるらしい。

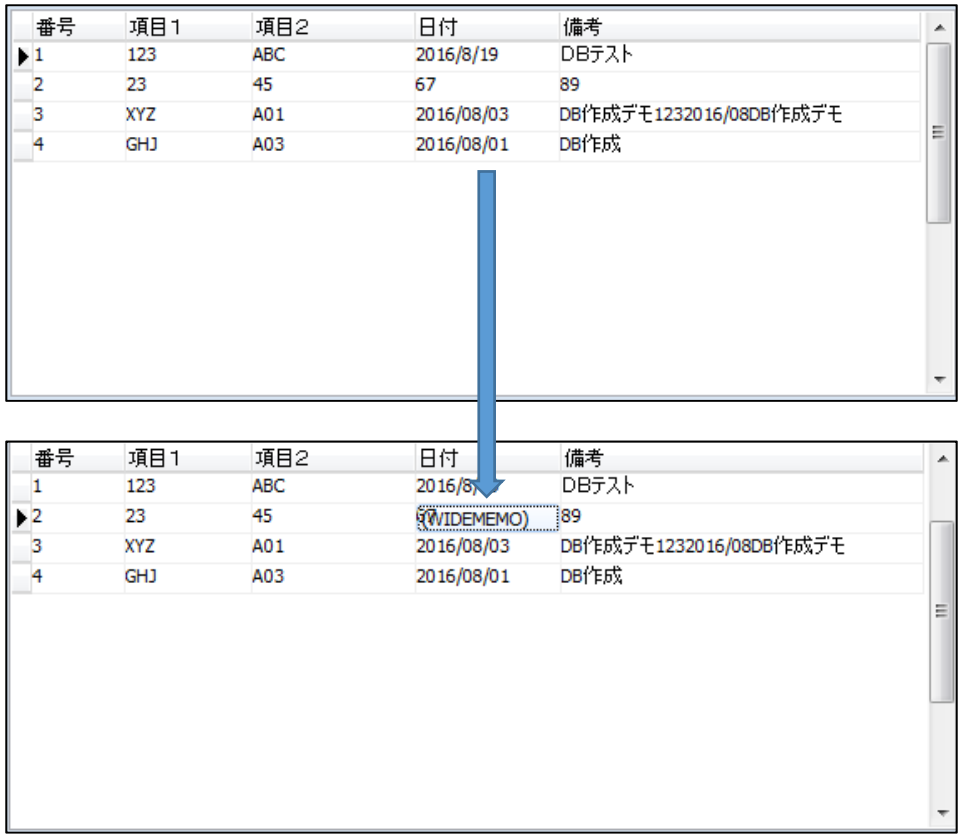
- OnGetTextイベントでテキストの値を変換する
- OnDrawColumnCellイベントで値を変換する

OnGetTextイベントでやるのが一般的なようだけれど、いや、そのイベントどこのコンポーネントにあるのかわからないんだけど。しょうがないのOnDrawColumnCellイベントの方でやることにする（このイベントはDBGridのイベントでに存在する）。で、コードは適当に外国のサイトにあるのを参考にしてみた。

```
procedure TmainForm.DBGrid1DrawColumnCell(Sender:
TObject; const Rect: TRect;
DataCol: Integer; Column: TColumn; State:
TGridDrawState);
begin
if Column.Field.FieldName = 'comment' then begin
DBGrid1.Canvas.FillRect(Rect);
DBGrid1.Canvas.TextRect(Rect, Rect.Left, Rect.Top,
Column.Field.Value);
end;
end;
```

4）FireDac（SQLite）ネットでの回避の仕方
ネットの回避の仕方で試してみる　OnDrawColumnCellイベントの方法1

- procedure TForm1.DBGrid1DrawColumnCell(Sender: TObject; const Rect: TRect;
- DataCol: Integer; Column: TColumn; State: TGridDrawState);
- begin
- DBGrid1.Canvas.FillRect(Rect);
- DBGrid1.Canvas.TextRect(Rect, Rect.Left, Rect.Top, Column.Field.Value);
- end;



番号	項目1	項目2	日付	備考
1	123	ABC	2016/8/19	DBテスト
2	23	45	67	89
3	XYZ	A01	2016/08/03	DB作成デモ1232016/08DB作成デモ
4	GHJ	A03	2016/08/01	DB作成

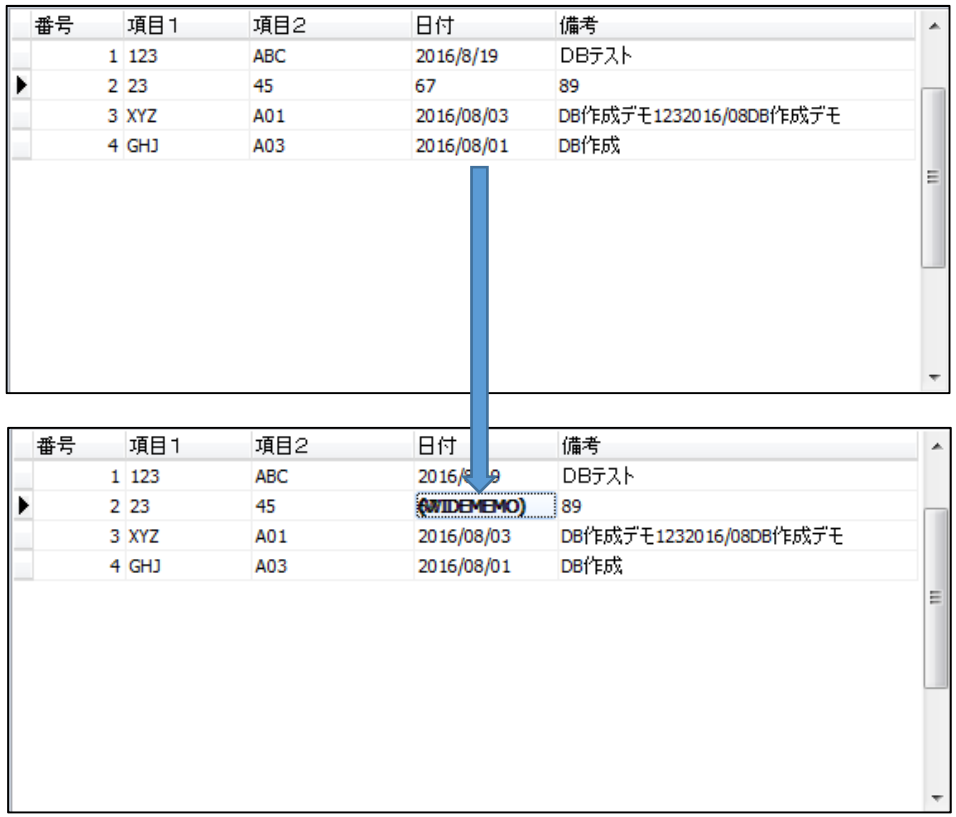
番号	項目1	項目2	日付	備考
1	123	ABC	2016/8/	DBテスト
2	23	45	(WIDEMEMO)	89
3	XYZ	A01	2016/08/03	DB作成デモ1232016/08DB作成デモ
4	GHJ	A03	2016/08/01	DB作成

文字が左詰めになって
しまうが、全部見える
やった！

しかしクリックすると
なんと（WIDEMEMO）が現れる。
残念！

4）FireDac（SQLite）ネットでの回避の仕方
ネットの回避の仕方で試してみる　OnDrawColumnCellイベントの方法2

```
inherited;
  DBGrid1.Canvas.FillRect(Rect);      //消す （WIDEMEMO)を消す
  DBGrid1.DefaultDrawColumnCell(Rect,DataCol,Column,State);// --- 描画を行う
  //(WIDEMEMO)の時
  if Column.Field.IsBlob then begin
    DBGrid1.Canvas.TextRect(Rect,Rect.left+2,Rect.Top+2,Column.Field.AsString );
  end;
```



WidoMemo表示のところだけ
にすると、表示はOK

やっぱりクリックすると
なんと（WIDEMEMO）が現れる。
残念！！！
見てくれが悪い。

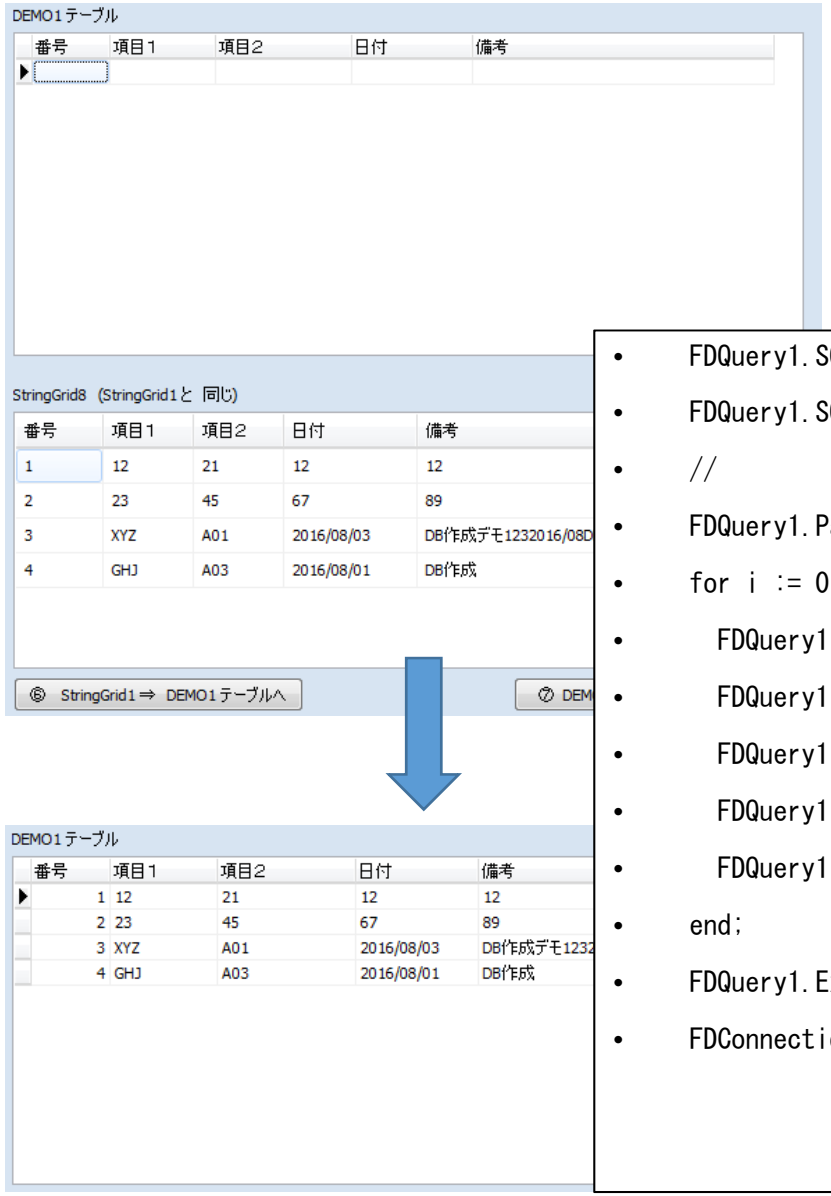
４）FireDac（SQLite）でのレコード数取得
DBをDataGridで表示させる　だめだめ問題５

- レコード数が大き時FDQuery1.RecordCountで取得すると50と表示される
- ⇒　何かかます、時間をおく
- ⇒　FDQuery1.FindLast;　正常に表示

```
FDQuery1.Close;
  FDQuery1.Open('select * from  DEMO1
ORDER by 番号 '+junban);
Edit6.Text:=inttostr(FDQuery1.RecordCount);
```

```
FDQuery1.Open('select * from  DEMO1 ORDER
by 番号 '+junban);
  FDQuery1.FindLast;
  Edit6.Text:=inttostr(FDQuery1.RecordCount);
  FDQuery1.FindFirst;
```

4）StringGrid ⇒ SQLite



FireDacへのデータ入力は
もっぱらDMLを使っています
（理由）これが一番速い！？

```
FDQuery1.SQL.Clear;
FDQuery1.SQL.Text := 'insert into DEMO1 values (:p0, :p1, :p2, :p3, :p4)';
//
FDQuery1.Params.ArraySize := StringGrid8.RowCount-1;
for i := 0 to FDQuery1.Params.ArraySize - 1 do begin
  FDQuery1.Params[0].AsIntegers[i] := strtoint(StringGrid8.Cells[0,i+1]);
  FDQuery1.Params[1].AsStrings[i]  := StringGrid8.Cells[1,i+1];
  FDQuery1.Params[2].AsStrings[i]  := StringGrid8.Cells[2,i+1];
  FDQuery1.Params[3].AsStrings[i] := StringGrid8.Cells[3,i+1];
  FDQuery1.Params[4].AsStrings[i] := StringGrid8.Cells[4,i+1];
end;
FDQuery1.Execute(FDQuery1.Params.ArraySize);
FDConnection1.Commit;
```

4）StringGrid ⇒ SQLite
配列DML、SQLite入門

表示設定

配列 DML コマンドのパフォーマンス（FireDAC）

コマンドの操作（FireDAC）への移動

このトピックでは、FireDAC でサポートしている配列 DML 機能のパフォーマンスの高さについて説明します。最初のこのトピックでは、簡単な例を使って、ほんの数行のコードを書くだけで 1 秒間に何千行ものレコードを挿入する方法を示します。

はじめに

目次 [非表示]
1 はじめに
2 テスト環境の準備方法
3 配列 DML コマンドの主な要素
4 使い方のヒント
5 配列 DML をテスト実行したときの典型的な結果
6 パフォーマンスのヒント
7 関連項目

FireDAC には配列 DML コマンドをデータベース サーバーごとに実装したものがすべてカプセル化されているため、ユーザーはサーバーの種類に関わらず同じコードを使用することができます。その結果のパフォーマンスは、サーバー実装によって異なります。主に Oracle や Microsoft SQL Server や IBM DB2 は配列 DML を非常に強力にサポートしているため、結果のパフォーマンスは目に見えて向上します。

まずサンプル コードを使用して、アプリケーションやネットワークのパフォーマンスを向上できる可能性を感じてみてください。

テスト環境の準備方法

以下の例では、FireDAC のサンプル データベース環境を使用します。このデータベースのインストール方法の詳細は、FireDAC デモ データベースを確認してください。デモ プロジェクトはサンプル ディレクトリに含まれています。

- このチュートリアルで使用するコード: FireDAC¥Samples¥Comp Layer¥TFDQuery¥ExecSQL¥AD03-ArrayDML
- 基本のコード例: FireDAC¥Samples¥Comp Layer¥TFDQuery¥ExecSQL¥Batch

配列 DML コマンドの主な要素

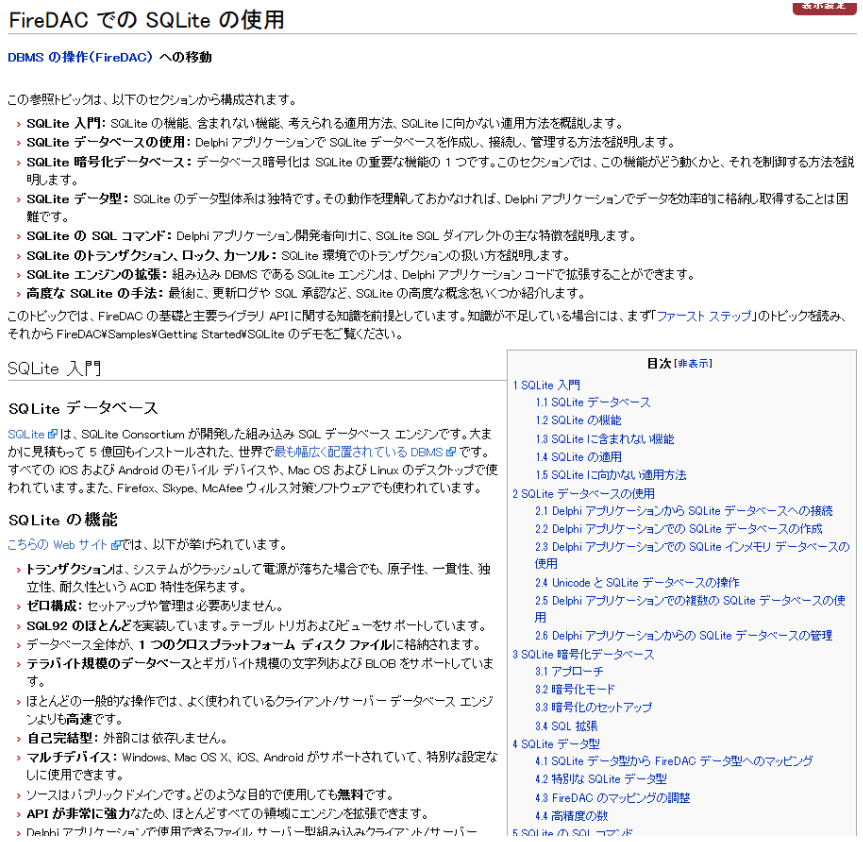
INSERT、UPDATE、DELETE など、パラメータを取るコマンドを N 回（通常は 1 つのレコードに対して 1 コマンド）実行しなければならない "使用例" を考えてください。この場合、1 組の入力パラメータごとに SQL コマンドの実行が要求され、そのたびにクライアントとサーバーの間でパラメータ群が転送されるため、ネットワークとクライアントとサーバーに高い負荷がかかります。

配列 DML を使用すると、一度の転送で 1 組だけではなく N 組のデータ群を送ることができます。次の例を見てください。

```
FDQuery1.SQL.Text:= 'insert into ADQA_Batch_test (tint, tstring) values(:f1, :f2)';
```

配列 DML コマンドを使用することで、コードを劇的に高速化できます。配列 DML コマンドでは、1 組だけではなく N 組のパラメータ群が転送されます。

```
FDQuery1.Params.ArraySize := 100;
...
for i := 0 to FDQuery1.Params.ArraySize do begin
  FDQuery1.Params[0].AsIntegers[i] := i;
  FDQuery1.Params[1].AsStrings[i] := 'Test' + IntToStr(i);
end;
FDQuery1.Execute(FDQuery1.Params.ArraySize);
```



FireDAC での SQLite の使用

DBMS の操作（FireDAC）への移動

この参照トピックは、以下のセクションから構成されます。

- **SQLite 入門:** SQLite の機能、含まれない機能、考えられる適用方法、SQLite に向かない適用方法を概説します。
- **SQLite データベースの使用:** Delphi アプリケーションで SQLite データベースを作成し、接続し、管理する方法を説明します。
- **SQLite 暗号化データベース:** データベース暗号化は SQLite の重要な機能の 1 つです。このセクションでは、この機能がどう動くかと、それを制御する方法を説明します。
- **SQLite データ型:** SQLite のデータ型体系は独特です。その動作を理解しておかなければ、Delphi アプリケーションでデータを効率的に格納し取得することは困難です。
- **SQLite の SQL コマンド:** Delphi アプリケーション開発者向けに、SQLite SQL ダイアレクトの主な特徴を説明します。
- **SQLite のトランザクション、ロック、カーソル:** SQLite 環境でのトランザクションの扱い方を説明します。
- **SQLite エンジンの拡張:** 組み込み DBMS である SQLite エンジンは、Delphi アプリケーション コードで拡張することができます。
- **高度な SQLite の手法:** 最後に、更新ログや SQL 承認など、SQLite の高度な概念をいくつか紹介します。

このトピックでは、FireDAC の基礎と主要ライブラリ API に関する知識を前提としています。知識が不足している場合には、まず「ファースト ステップ」のトピックを読み、それから FireDAC¥Samples¥Getting Started¥SQLite のデモをご覧ください。

SQLite 入門

目次 [非表示]
1 SQLite 入門
1.1 SQLite データベース
1.2 SQLite の機能
1.3 SQLite に含まれない機能
1.4 SQLite の適用
1.5 SQLite に向かない適用方法
2 SQLite データベースの使用
2.1 Delphi アプリケーションから SQLite データベースへの接続
2.2 Delphi アプリケーションでの SQLite データベースの作成
2.3 Delphi アプリケーションでの SQLite インメモリ データベースの使用
2.4 Unicode と SQLite データベースの操作
2.5 Delphi アプリケーションでの複数の SQLite データベースの使用
2.6 Delphi アプリケーションからの SQLite データベースの管理
3 SQLite 暗号化データベース
3.1 アプローチ
3.2 暗号化モード
3.3 暗号化のセットアップ
3.4 SQL 拡張
4 SQLite データ型
4.1 SQLite データ型から FireDAC データ型へのマッピング
4.2 特別な SQLite データ型
4.3 FireDAC のマッピングの調整
4.4 高精度の数
5 SQLite の SQL コマンド

SQLite データベース

SQLite は、SQLite Consortium が開発した組み込み SQL データベース エンジンです。大まかに見積もって 5 億回もインストールされた、世界で最も幅広く配置されている DBMS です。すべての iOS および Android のモバイル デバイスや、Mac OS および Linux のデスクトップで使われています。また、Firefox、Skype、McAfee ウィルス対策ソフトウェアでも使われています。

SQLite の機能

こちらの Web サイトでは、以下が挙げられています。

- **トランザクション**は、システムがクラッシュして電源が落ちた場合でも、原子性、一貫性、独立性、耐久性という ACID 特性を保ちます。
- **ゼロ構成:** セットアップや管理は必要ありません。
- **SQL92 のほとんど**を実装しています。テーブル トリガおよびビューをサポートしています。
- データベース全体が、**1 つのクロスプラットフォーム ディスク ファイル**に格納されます。
- **テラバイト規模のデータベース**とギガバイト規模の文字列および BLOB をサポートしています。
- ほとんどの一般的な操作では、よく使われているクライアント/サーバー データベース エンジンよりも**高速**です。
- **自己完結型:** 外部には依存しません。
- **マルチデバイス:** Windows、Mac OS X、iOS、Android がサポートされていて、特別な設定なしに使用できます。
- ソースはパブリック ドメインです。どのような目的で使用しても**無料**です。
- **API が非常に強力**なため、ほとんどすべての領域にエンジンを拡張できます。
- Delphi アプリケーションで使用できるファイル サーバー型組み込みクライアント/サーバー

4） SQLite ⇒ StringGrid

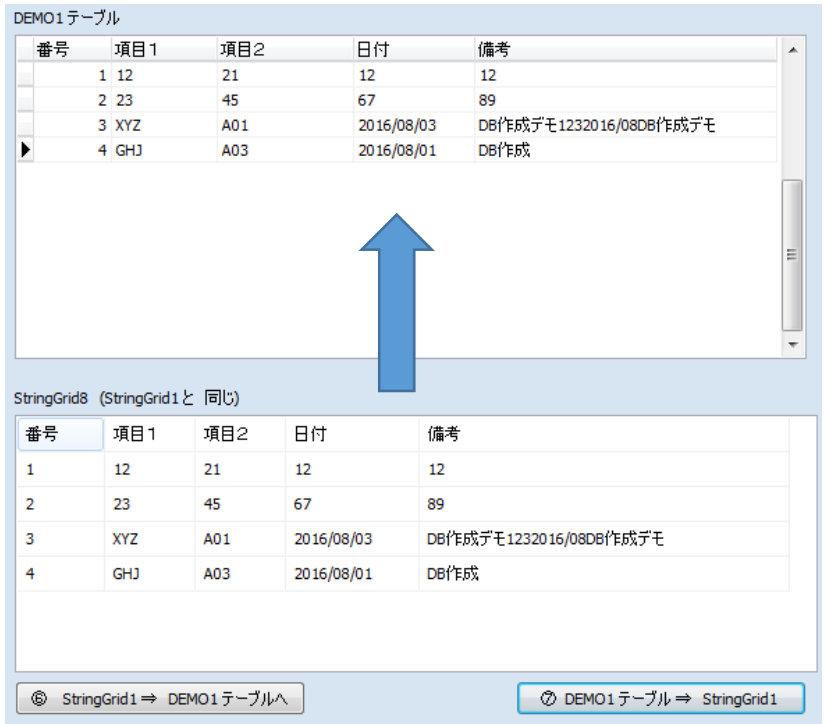
レコード数取得：FDTable1.RecordCount;

項目数取得　　：FDQuery1.FieldCount;

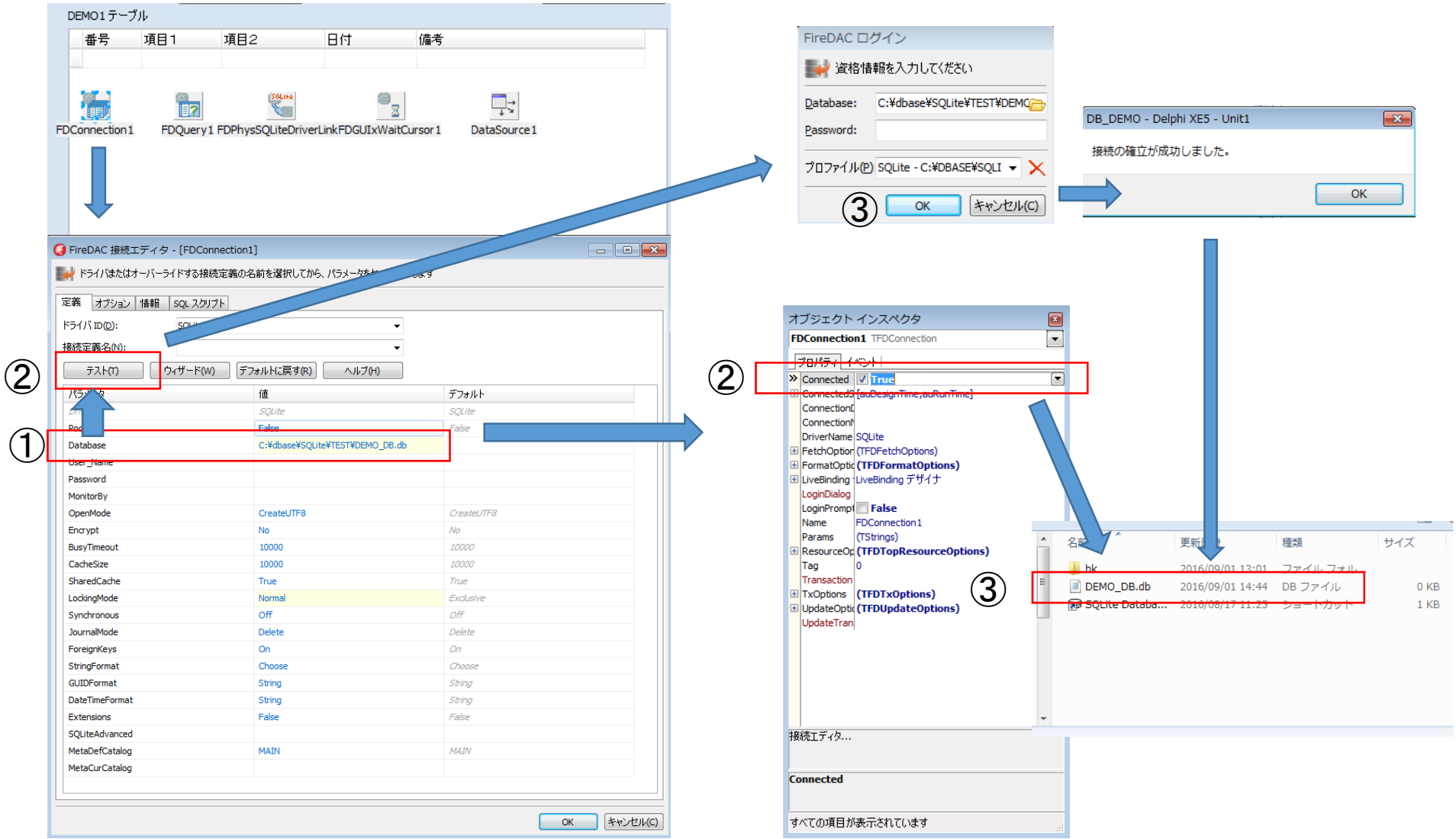
フィールド値取得：FDQuery1.Fields[j].AsString;

レコードのコントロール：FDQuery1.FindFirst;

レコードのコントロール：FDQuery1.Next;



4)もっと簡単に。。。その1：（今回のプレゼン用に作成してみました）
Delphiからデータベースファイル作成



お好きな方法でデータベースファイルを作成する

4)もっと簡単に。。。その2： 作業の流れ
大分かんたんになったと思います

①

StringGrid8 (StringGrid1と 同じ)　① EXCELファイル読込

番号	項目1	項目2	日付	備考
1	12	21	12	12
2	23	45	67	89
3	XYZ	A01	2016/08/03	DB作成デモ1232016/08DB作成デモ
4	GHJ	A03	2016/08/01	DB作成

②　② DBファイルはFDConnection1で作成

③　③ StringGrid1 ⇒ テーブル作成用SQL

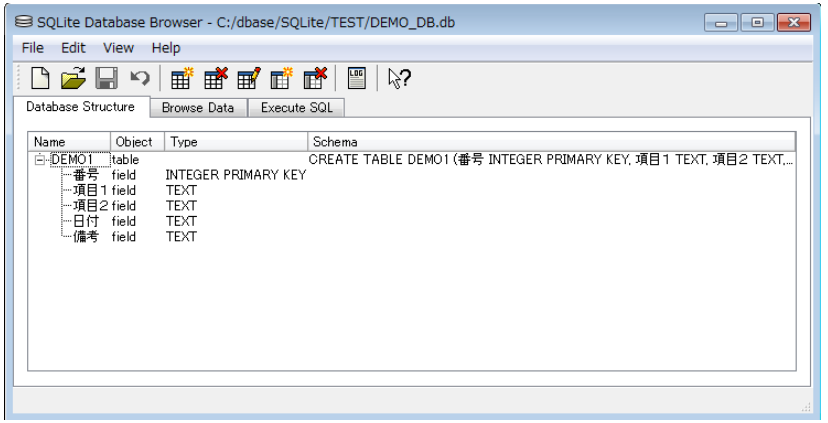
③ テーブル作成用SQL = create table Shinsei + 番号 INTEGER PRIMARY KEY, 項目1 TEXT, 項目2 TEXT, 日付 TEXT, 備考 TEXT

④　④テーブル作成用SQL ⇒ DEMO1テーブル作成

⑤　⑤ StringGrid1 ⇒ DEMO1テーブルへ

⑥　⑥ DEMO1テーブル ⇒ StringGrid1

⑦　⑦ Excelファイル出力



DEMO1テーブル

番号	項目1	項目2	日付	備考
1	12	21	12	12
2	23	45	67	89
3	XYZ	A01	2016/08/03	DB作成デモ1232016/08DB作成デモ
4	GHJ	A03	2016/08/01	DB作成

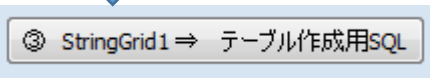
Excelの項目とそのデータがSQLになった！

4)もっと簡単に。。。その4:

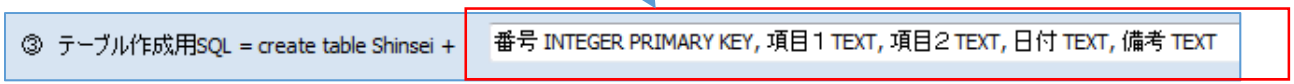
①

②

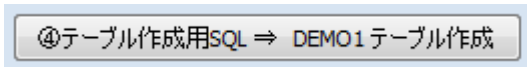
③



手で編集後



④



①
```
begin
 GetDir(0,DIR);
 Filename:=DIR+'¥'+Edit1.Text;       //Excel File
 St:=1;                              //Sheet 番号
 Excel_SG(StringGrid1,St,Filename,Err);  //Excel to StringGrid
End;
```

③
```
var
 i:integer;
 s:string;
begin
 s:='番号 INTEGER PRIMARY KEY';
 for i := 0 to StringGrid8.ColCount - 1 do begin
  s:=s+', '+StringGrid8.Cells[i,0]+' TEXT';
 end;
 Edit16.Text:=s;
end;
```

④
```
begin
 FDQuery1.SQL.Clear;
 FDQuery1.SQL.Add('create table DEMO1 ('+Edit16.text+');');
 FDQuery1.ExecSQL;
end;
```

４）もっと簡単に。。。その４： コーディングDMLこの部分をN対応にしたいが。。。

⑤ 


```
var
  i:integer;
begin
  //
  //
  FDQuery1.SQL.Clear;
  FDQuery1.SQL.Add('delete from DEMO1');  //
  FDQuery1.ExecSQL;
  //
  //
  FDQuery1.SQL.Clear;
  FDQuery1.SQL.Text := 'insert into DEMO1 values (:p0, :p1, :p2, :p3, :p4)';
  //
  FDQuery1.Params.ArraySize := StringGrid8.RowCount-1;
  for i := 0 to FDQuery1.Params.ArraySize - 1 do begin
    FDQuery1.Params[0].AsIntegers[i] := strtoint(StringGrid8.Cells[0,i+1]);
    FDQuery1.Params[1].AsStrings[i]  := StringGrid8.Cells[1,i+1];
    FDQuery1.Params[2].AsStrings[i]  := StringGrid8.Cells[2,i+1];
    FDQuery1.Params[3].AsStrings[i] := StringGrid8.Cells[3,i+1];
    FDQuery1.Params[4].AsStrings[i] := StringGrid8.Cells[4,i+1];
  end;
  FDQuery1.Execute(FDQuery1.Params.ArraySize);
  FDConnection1.Commit;
  //
  FDQuery1.Close;
  FDQuery1.Open('select * from DEMO1');
  //
end;
```

4）もっと簡単に。。。その5： もっと簡単に。。。願いを込めて

⑥


```
var
 s:string;
 i,j:integer;
begin
 //
 // 項目名取得
 //
 Memo1.Clear;
 FDQuery1.Close;
 FDQuery1.Open('select * from  DEMO1');
 for i := 0 to FDQuery1.FieldCount -1 do begin
  Memo1.lines.Add(FDQuery1.Fields[i].FieldName);
 end;
 //
 //
 FDQuery1.FindFirst;
 StringGrid1.RowCount:=FDQuery1.RecordCount+1;
 StringGrid1.ColCount:=FDQuery1.FieldCount;
 //
 // 項目名セット
 //
 for i := 0 to Memo1.lines.count -1 do begin
  StringGrid1.cells[i,0]:=Memo1.lines[i];
 end;
 //
 for i := 1 to FDQuery1.RecordCount do begin
  for j := 0 to Memo1.lines.count -1 do begin
   s:=FDQuery1.Fields[j].AsString;
   StringGrid1.cells[j,i]:=s;
  end;
  if not (i = FDQuery1.RecordCount) then begin
   FDQuery1.Next;
  end;
 end;
 StringGrid1.RowCount:=i;
 if  i > 1 then StringGrid1.FixedRows:=1;
 //
end;
```

⑦


```
var
 Filename:string;
begin
 Filename:=Edit2.Text;
 ExportToExcel(StringGrid1,Filename); //Excel 出力
end;
```

4） SQLite 編集用、検索用、備忘録用

ようやくDatabaseDesktopを使っていた時と同じ環境を構築できた！

１）各項目はDBEditかDBMemoを利用

２）更新：FDQuery1.ApplyUpdates(0);

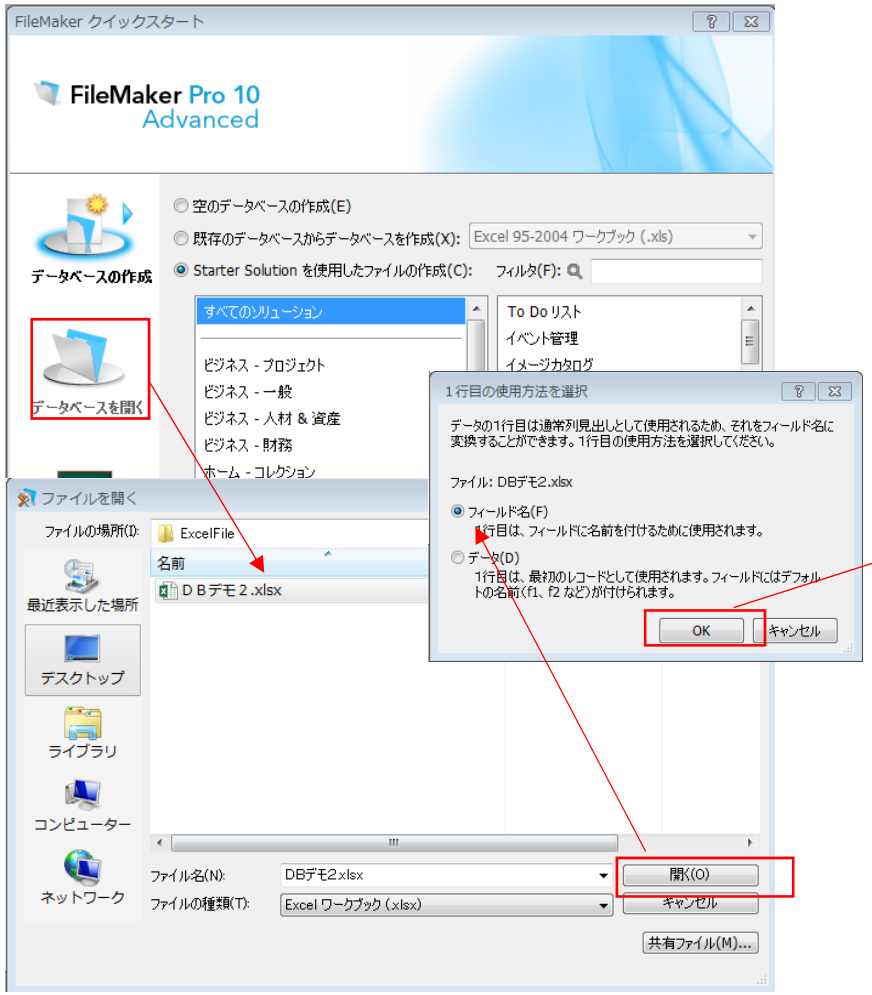
忘れがちなものを簡単にゲット

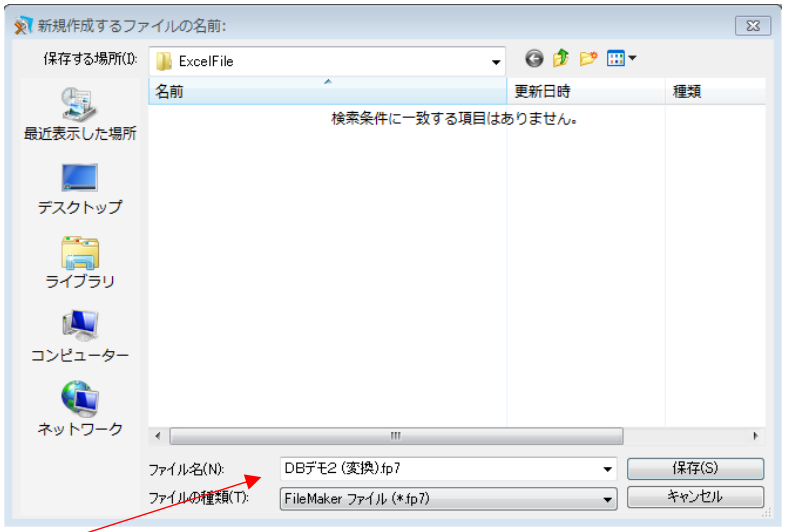
検索関係

SQL文はこれで簡単に確認３階層あればほぼＯＫ！

備忘：SQL文をすぐに忘れるので…ここでチェック！。

5） FileMakerProでレポート作成 ①形式変換





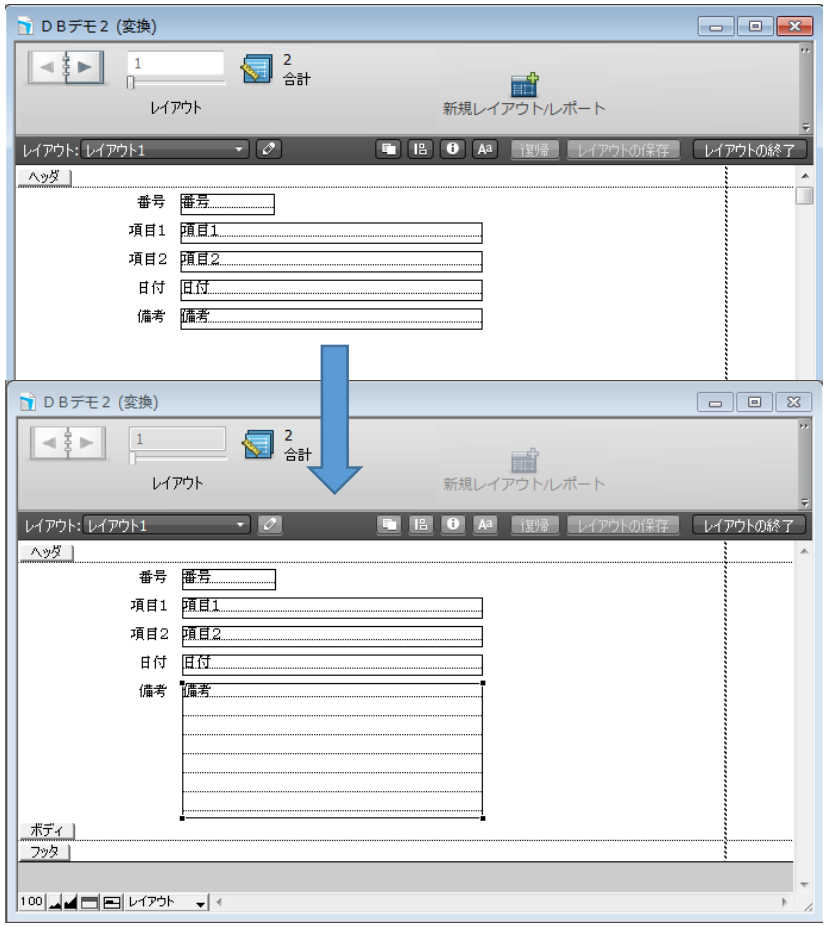
FileMakerProはレポートデザインをしながらDB項目を作れるので。。。！
FistRepotもあるが…。

堅苦しいことは言わすにもうファイルメーカとして作成される

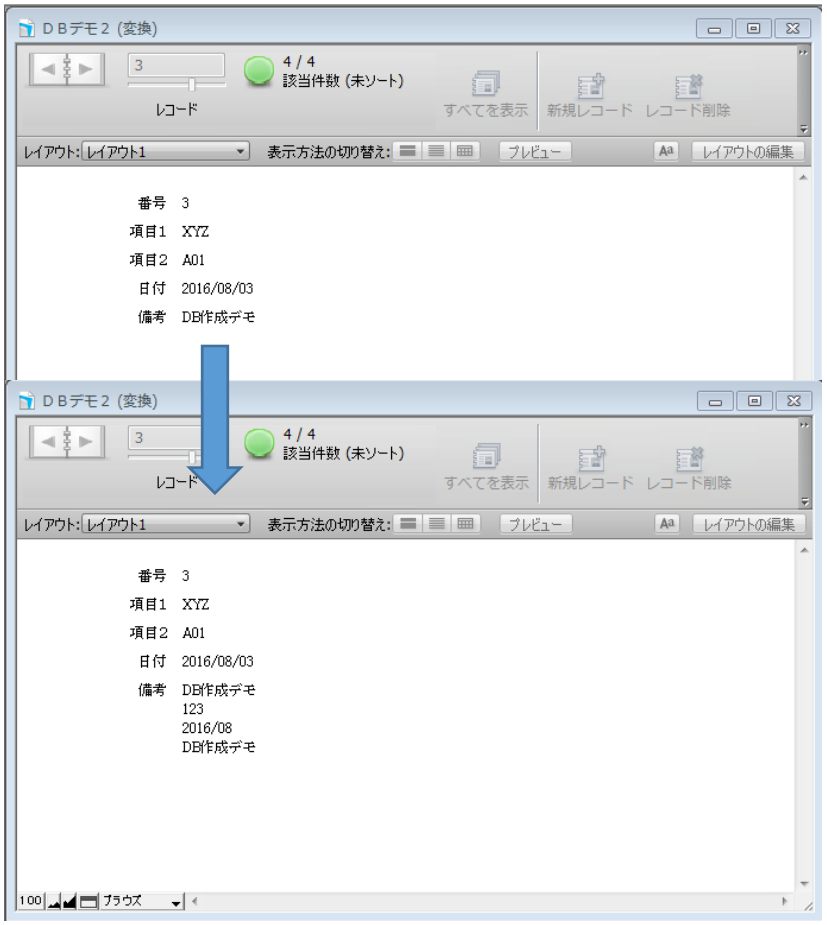
5） FileMakerProでレポート作成 ②レポート形式

レイアウト1：レポート形式　デフォルト

レイアウト編集



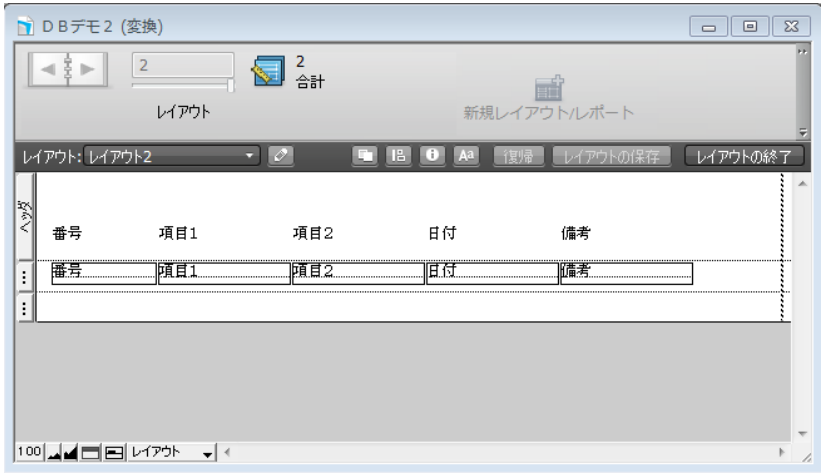
ブラウズ



5） FileMakerProでレポート作成 ②表形式

レイアウト2：表形式：デフォルト

レイアウト編集画面



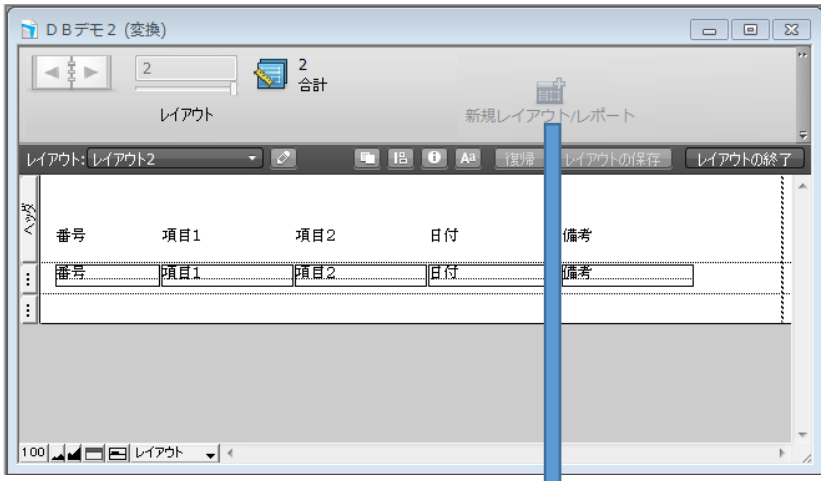
ブラウズ画面

番号	項目1	項目2	日付	備考
1	12	21	12	12
2	23	45	67	89
3	XYZ	A01	2016/08/03	DB作成デモ
4	GHJ	A03	2016/08/01	DB作成

5） FileMakerProでレポート作成 ②表形式

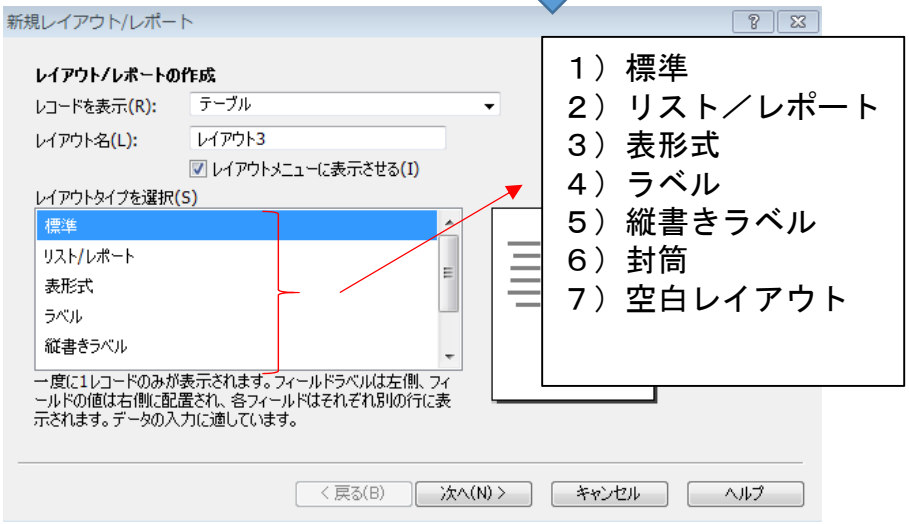
レイアウト2：表形式：デフォルト

レイアウト編集画面

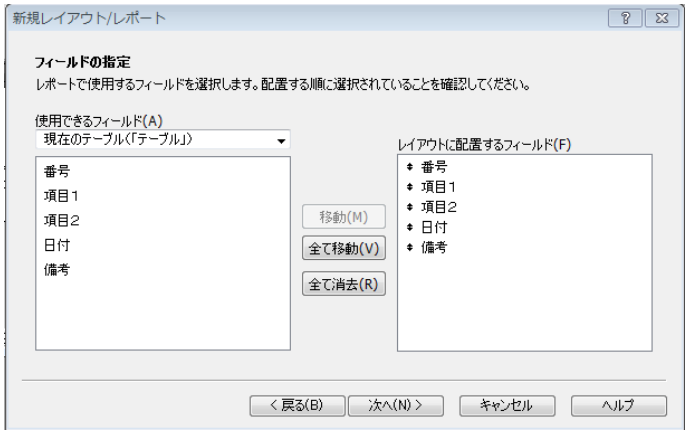


ブラウズ画面

レイアウトの追加

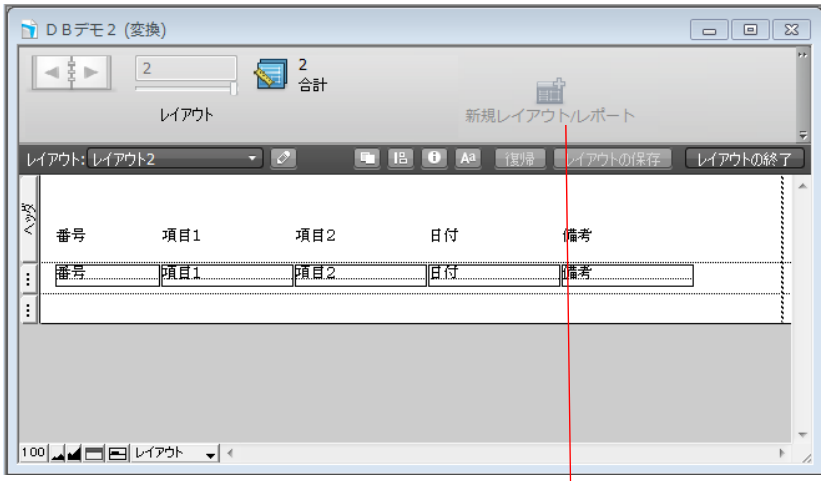


1） 標準
2） リスト／レポート
3） 表形式
4） ラベル
5） 縦書きラベル
6） 封筒
7） 空白レイアウト

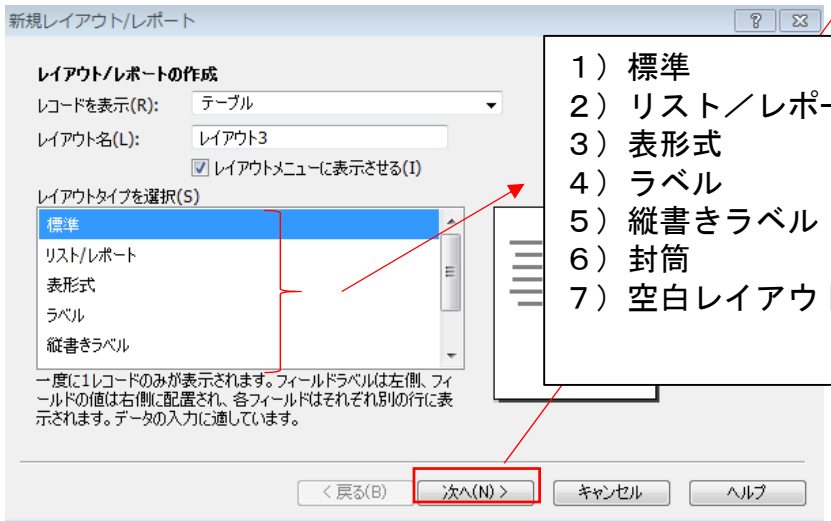


5） FileMakerProでレポート作成　③レイアウト追加

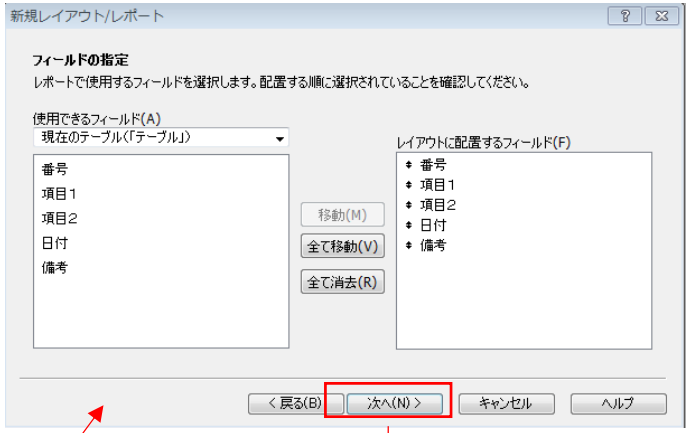
レイアウト編集

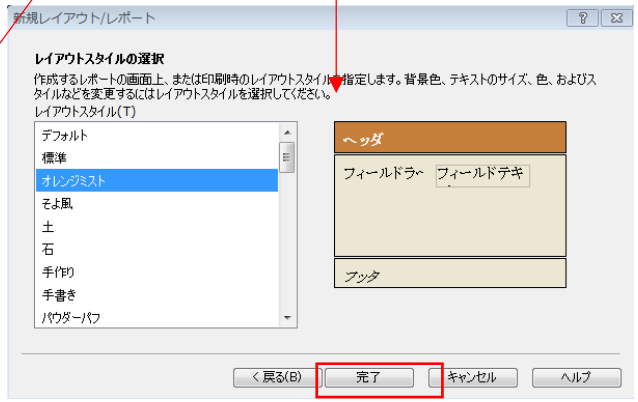


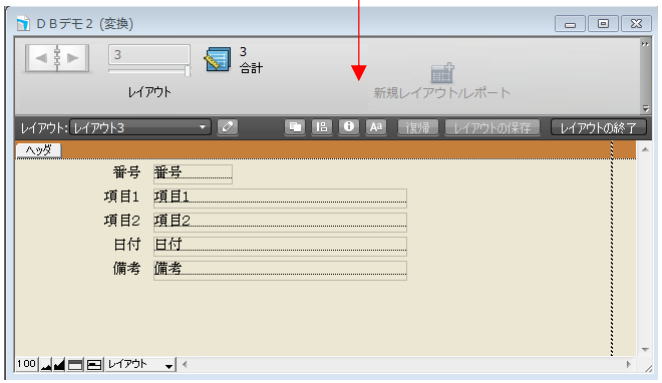
レイアウトの追加



1） 標準
2） リスト／レポート
3） 表形式
4） ラベル
5） 縦書きラベル
6） 封筒
7） 空白レイアウト





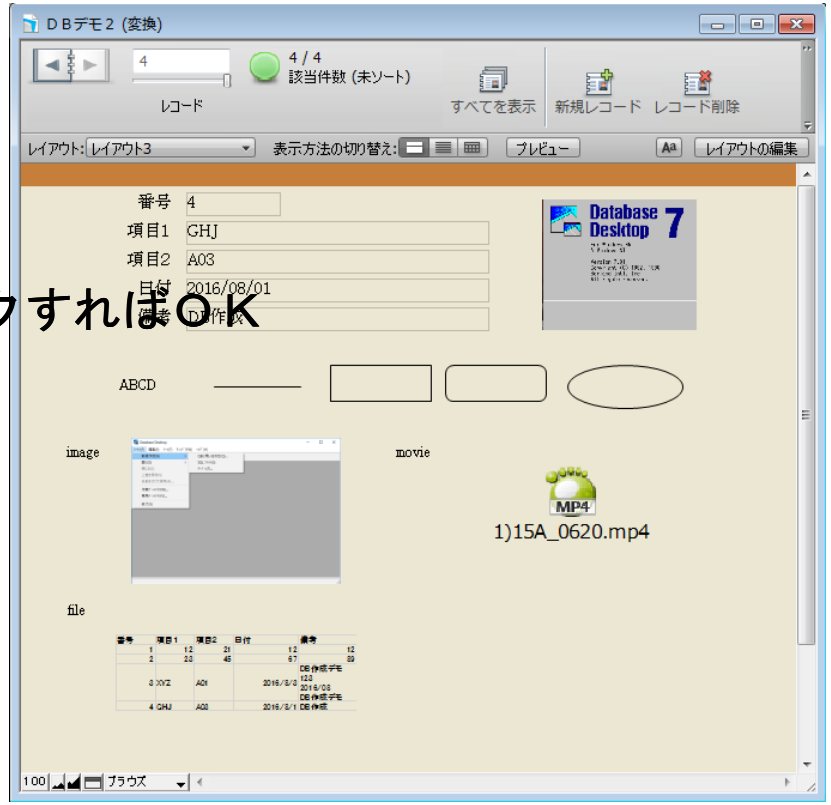


5） FileMakerProでレポート作成 ④レイアウト編集

ちょっとしたドロー機能
Excel、画像、動画もデータベースを編集すれば、

ファイルをドラックすればＯＫ

ファイルをドラックすればＯＫ



データベース設計と言うよりレポート設計をしている感じが好きです！

７） 最後に私の理想の設計方法になるにはと欲しいもの

- １）エクセルで設計開始
- ２）ファイルメーカでレポートデザイン設計
- ３）エクセルからデータベース作成
- ４）データベースからFileMakerにデータを渡してレポート表示

- 欲しいもの）
- １）StringGrid1.LoadFromExcelFile();Excelからデータ読込、StringGrid1.SaveToExcelFile();データExcel出力
- ２）DBGrid1.LoadFromExcelFile();Excelからデータ読込、DBGrid1.SaveToExcelFile();データExcel出力
- ３）FDPhysFileMakerDriverLink1；FireDACでFileMakerが扱える
- ４）多機能検索コンポーネント

ＰＳ）RichEditのバグとしょうもない質問

- Delphi RichEdit UTF-8 テキスト読み込み時文字化けが起こる
- ⇒TMemoがTStringListを使いましょう！
- Tmemoの容量制限が以前はあったが、今はないと以前のRAD Studio勉強会@Osakaで教えてもらったので、今はもっぱらTmemoでファイルを読み込んでいます。

```
var
SL : TStringList;
begin
  SL := TStringList.Create;
  try
    SL.LoadFromFile(Filename);
    for i := 0 to SL.Count-1 do begin
      RichEdit1.Lines.Add(SL[i]);
    end;
  finally
    SL.Free;
  end;
end;
```

しょうもない質問）

プログラムの終わらせ方
いつも安易に①ですが…
皆様は？

①
```
begin
  Halt;
end;
```

②
```
begin
    Form1.Close;
end;
```

ご清聴？有難うございました

ロードマップの大切ですが、以前のバージョンのバグの修正もしてほしいなぁ！
これからはC++BuilderStarterか！DelphiStarterか！