

# 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...

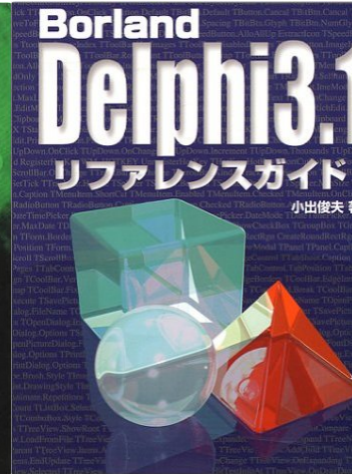
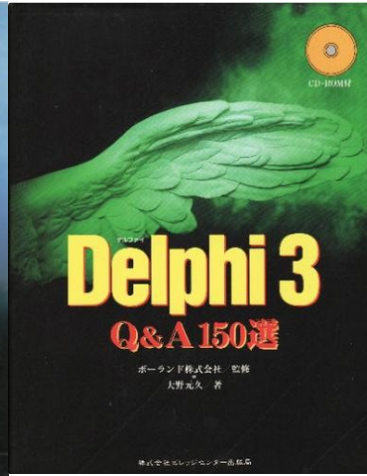
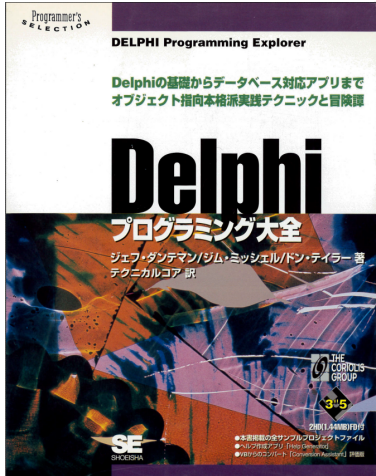
僕のDelphi入門書  
物語風の説明がお気に！

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...



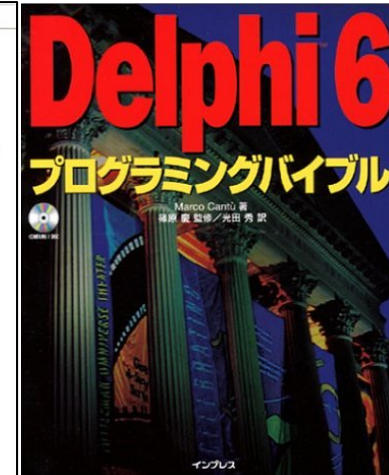
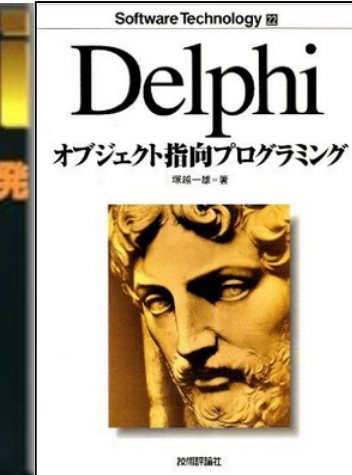
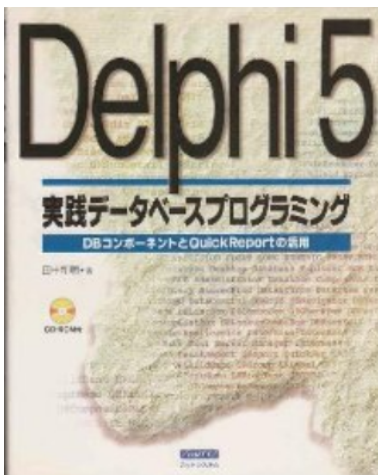
データベースはこれ？

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

はじめに .....	iv
付録ディスクについて .....	vi
著者紹介 .....	ix
<b>PART1 Delphiの得意分野 .....</b>	<b>1</b>
<b>Chapter1 RADでいこう！ .....</b>	<b>3</b>
Delphiのツール .....	4
メインウィンドウ .....	5
オブジェクトインスペクタ .....	6
フォームウィンドウ .....	6
ユニットウィンドウ .....	6
Delphiでのコーディング .....	6
コンポーネントをフォームに配置する .....	9
同じ種類のコンポーネントを複数個配置する .....	10
コンポーネントの移動とサイズ変更 .....	11
コンポーネントの位置合わせ .....	12
作業を保存する .....	14
プログラムの実行 .....	14
オブジェクトのプロパティを変更 .....	15
フォームを改良する .....	15
イベントへの応答 .....	17
これであなたも立派なプログラマ .....	19
<b>Chapter2 これでホントにプログラミング？ .....</b>	<b>21</b>
Delphiの形態 .....	22
すべてのものは見えるか見えないかである .....	23
刺激と応答 .....	24
ユニットライブラリ .....	26
どれだけのプログラミングが必要か .....	27
Delphiで独自のコンポーネントを作る .....	28
Pascalへの中思考型アプローチ .....	29
大思考対小思考 .....	30
中思考とインターフェース .....	31
中思考としてのDelphi .....	32
これはホントにプログラミングなのです .....	34

<b>Chapter3 コンポーネントとプロパティ .....</b>	<b>35</b>
コンポーネントのプロパティとは .....	36
フォームコンポーネント .....	37
フォームのサイズと位置を変更する .....	37
フォームの枠を変更する .....	38
その場で色を変える .....	39
セットとサブプロパティ .....	41
BorderIconsプロパティ .....	42
Fontプロパティ .....	43
親のプロパティを継承する .....	45
ParentCtl3D, ParentColor, ParentFont .....	45
OnCreate: コンポーネントプロパティを起動時に設定する .....	47
実行時プロパティと読み込み専用プロパティ .....	48
色の変更を反映する .....	50
チェックボックスの操作 .....	50
プログラムを保存し実行する .....	51
プロパティ管理 .....	52
<b>Chapter4 バケツに入れる .....</b>	<b>53</b>
変数: データを入れるバケツ .....	54
変数とその型 .....	55
バケツを作る .....	55
バケツを満たす .....	57
リテラル定数 .....	58
名前付き定数 .....	58
その他の数値型 .....	59
長整数 .....	59
浮動小数点数 .....	60
算術演算子 .....	62
式 .....	62
整数と浮動小数点数の混在 .....	63
除算 .....	63
ModとDivを使った整数の除算 .....	64
Comp型: 不思議な数値 .....	64
文字 .....	65
順序型 .....	66
標準関数 .....	66
コメント .....	67
文字列 .....	68
論理型: 真か偽か .....	70
論理値と関係演算子 .....	71
論理値とIF文 .....	72
複合文 .....	73
Object Pascalの基礎 .....	74

# 僕のDelphi入門書 Delphiプログラミング大全から

1行でレコードを挿入する .....	400
まだ終わっていただけではありません .....	405
<b>PART3 不誠実なデモ事件 エース・ブレイクポイントの事件簿より .....</b>	<b>407</b>
<b>Chapter15 プログラミングこそ天職なり .....</b>	<b>409</b>
主人公の紹介 .....	411
<b>Chapter16 現場検証 .....</b>	<b>421</b>
種あかし .....	423
リユートフィスクの家で .....	423
テーブルは回る .....	425
テーブル1:顧客テーブル .....	426
テーブル2:ベンダーテーブル .....	428
テーブル3:従業員テーブル .....	429
テーブル4:汎用ヒントテーブル .....	430
テーブル5:製品テーブル .....	431
テーブル6:インボイステーブル .....	434
テーブル7:インボイス品目テーブル .....	435
テーブル8:RMAテーブル .....	436
データテーブルの作成とデータの挿入 .....	439
クイック&ダーティなユーティリティ1の作成 .....	441
クイック&ダーティなユーティリティ2の作成 .....	443
<b>Chapter17 それはライブラリで起こった .....</b>	<b>445</b>
ステップ1:アプリケーションの範囲を確定 .....	447
ステップ2:アプリケーション機能の細分化 .....	447
ステップ3:アプリケーションのラフレイアウト .....	449
プロジェクトディレクトリの選択 .....	449
別名を使え .....	449
新規プロジェクトの作成 .....	451
ラフな製品情報フォーム .....	451
ラフなRMAマネージメントフォーム .....	454
RMAってどこから来るの? .....	456
ラフなRMA入力フォーム .....	457
バック・トゥ・ザ・RMA入力フォーム .....	459
さらに検討 .....	461
<b>Chapter18 プロパティの設定 .....</b>	<b>463</b>
ステップ4:各コンポーネントのプロパティ設定 .....	464
メインフォームの改訂 .....	468

製品情報フォームの更新 .....	470
RMAマネージメントフォームの改良 .....	474
RMA入力フォームの改訂 .....	479
初めてのコンパイルと実行 .....	483
<b>Chapter19 意外な展開 .....</b>	<b>487</b>
ステップ5:イベントハンドラの作成 .....	488
メインフォームのイベント処理 .....	488
製品情報フォームのイベント処理 .....	489
RMAフォーム第1弾 .....	496
RMAフォーム第2弾 .....	504
NextRMA番号と日付 .....	505
インボイステーブルのレコード .....	505
ComboBoxからの情報 .....	506
指定されたRMAおよびインボイスレコードに移る .....	508
指定された品目レコードを編集モードに .....	509
ComboBoxにイニシャルを .....	510
ComboBoxに状態を .....	511
RMA入力フォームの仕上げ .....	511
RMAマネージメントフォームの仕上げ .....	520
2つの新しいルーチン .....	520
[Delete] ボタンのイベントハンドラ .....	521
個人的告白 .....	526
<b>Chapter20 現場へ戻る .....</b>	<b>527</b>
ステップ6:その1:見栄えのよい製品情報フォーム .....	528
ステップ6:その2:RMA入力フォームへの追加 .....	534
日付の更新 .....	534
コンボボックスにイニシャルを追加 .....	534
インボイスの検索 .....	535
RMA番号のフォーマット .....	538
状態メッセージの改良 .....	539
RMA入力フォームの仕上げ .....	541
脱力感 .....	551
ステップ6:その3:RMAマネージメントフォームの改良 .....	553
RMAを番号で検索 .....	554
削除の確認 .....	555
ボタンの有効と無効 .....	556
状態メッセージ定数を組み込む .....	558
RMAマネージメントフォームをみがき上げる .....	558
ステップ6:その4:メインフォームの改良 .....	559
決心 .....	564

# 僕のDelphi入門書 Delphiプログラミング大全から

**RADでいこう!**



ジム・ミッシェル

Delphiの目的は、迅速なアプリケーションの開発 (Rapid Application Development, RAD)です。

**プログラミングこそ  
天職なり**



ドン・テイラー

エース・ブレイクポイントは非伝統的プログラミングの大冒険を始める。冒険は、プロジェクトの印象を把握することから始まる。

# 僕のDelphi入門書 Delphiプログラミング大全から

PART3 不誠実なデモ事件 ―エース・ブレイクポイントの事件簿より

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速い自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きましょう。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、パウルスポ(Paulsbo)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

プログラミングこそ天職なり

であるといっただけでしょう。マリナーや、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、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・ブレイクポイント、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術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プログラマーであった。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。

パウルスポに移り住み、スクウェアミッシュ・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。

たいいてい主人公同様、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と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。まちがいの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、巧みに身をかかわりながら、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、断じてそうであるはずありません。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、スクリーンにイスを近づけて、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・ブレイクポイントの事件簿より

### 太平洋岸コミュニティにさんざんと輝く星, 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したとき、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。マイヤーズ夫婦は、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、ワシントン州パウルスポに移転します。1989年の11月でした。新しいオフィスはリパ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1千万ドルの大白に乗ります。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？ マイヤー夫婦は、テクノロジーに遅れをとらなってきたことの他に、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時間出荷することを保証することでも有名です。

マイヤー夫婦の願望は、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るうちは、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音が窓を打っている。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・シューブリームとが、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つ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・ブレイクポイントです。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ぞ、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・ブレイクポイントです。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 :もしかしていませんか？
- 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～

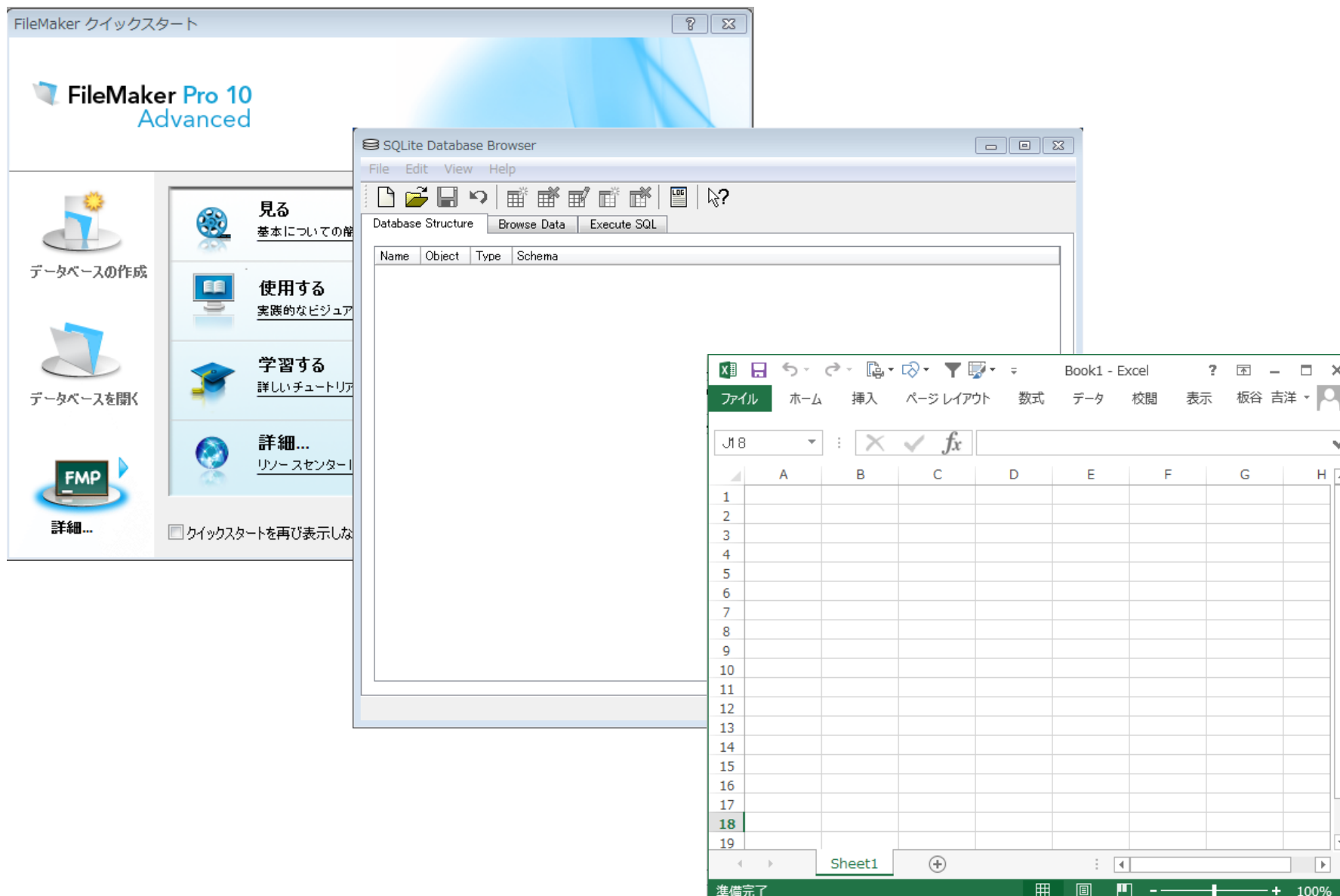
## 2) FileMakerPro : 私の一番のお気に入り いいところ

- 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...)

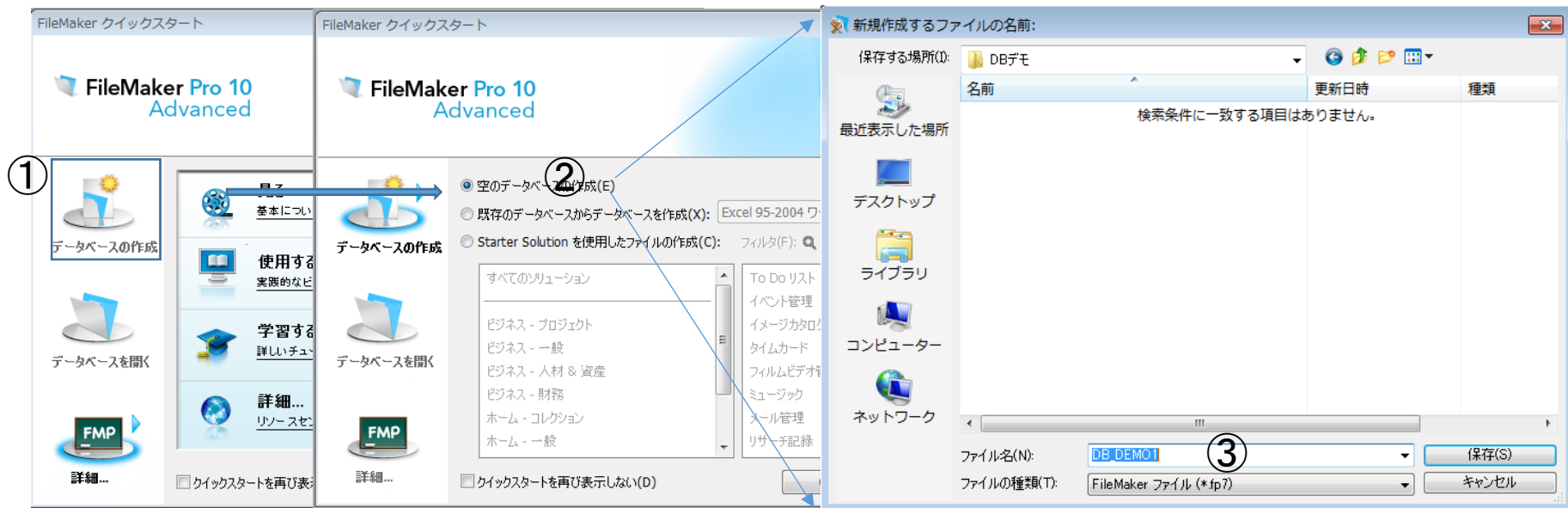
## 2) FileMakerPro : 私の一番のお気に入り だめだめなところ

- 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つとしない。

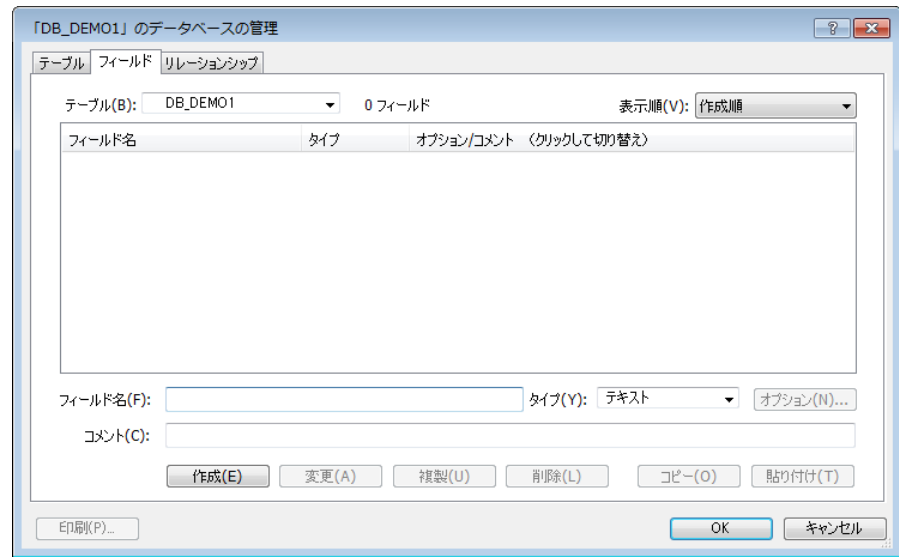
# 3) かんたんデータベース作成



### 3) 例えば FileMakerPro等のDBでのノーマル設計



④

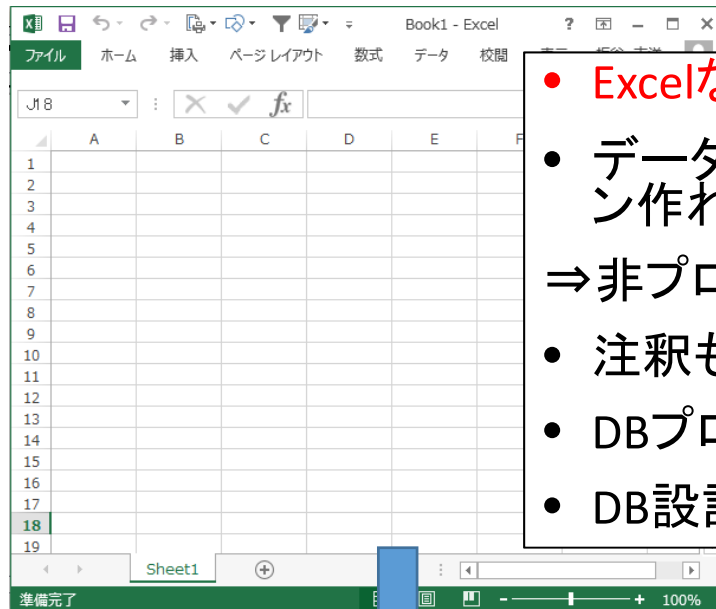


⑤

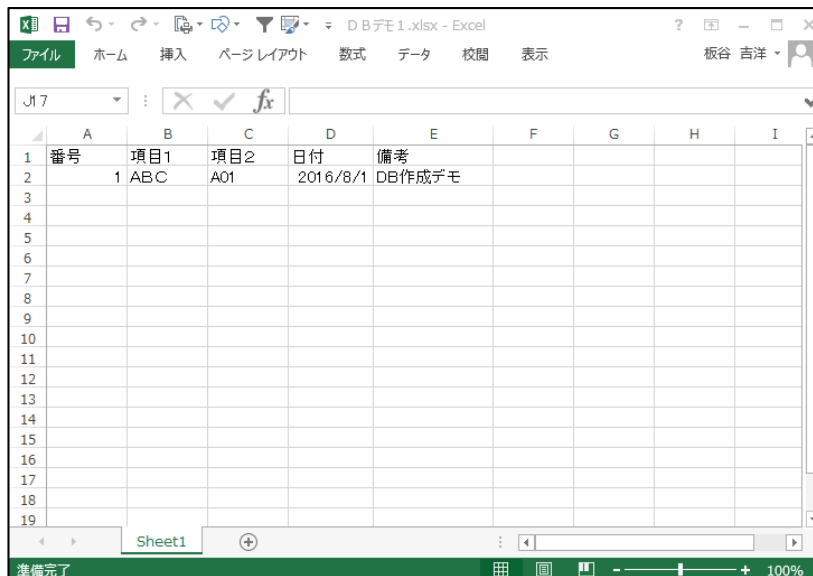
大体いつも考えていなの  
でいきなり挫折！  
全体を考えずに作りながら  
決めていくので。。。  
典型的なスパゲッティープ  
ログラマーです

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
- 注釈もががが書き込める
  - 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

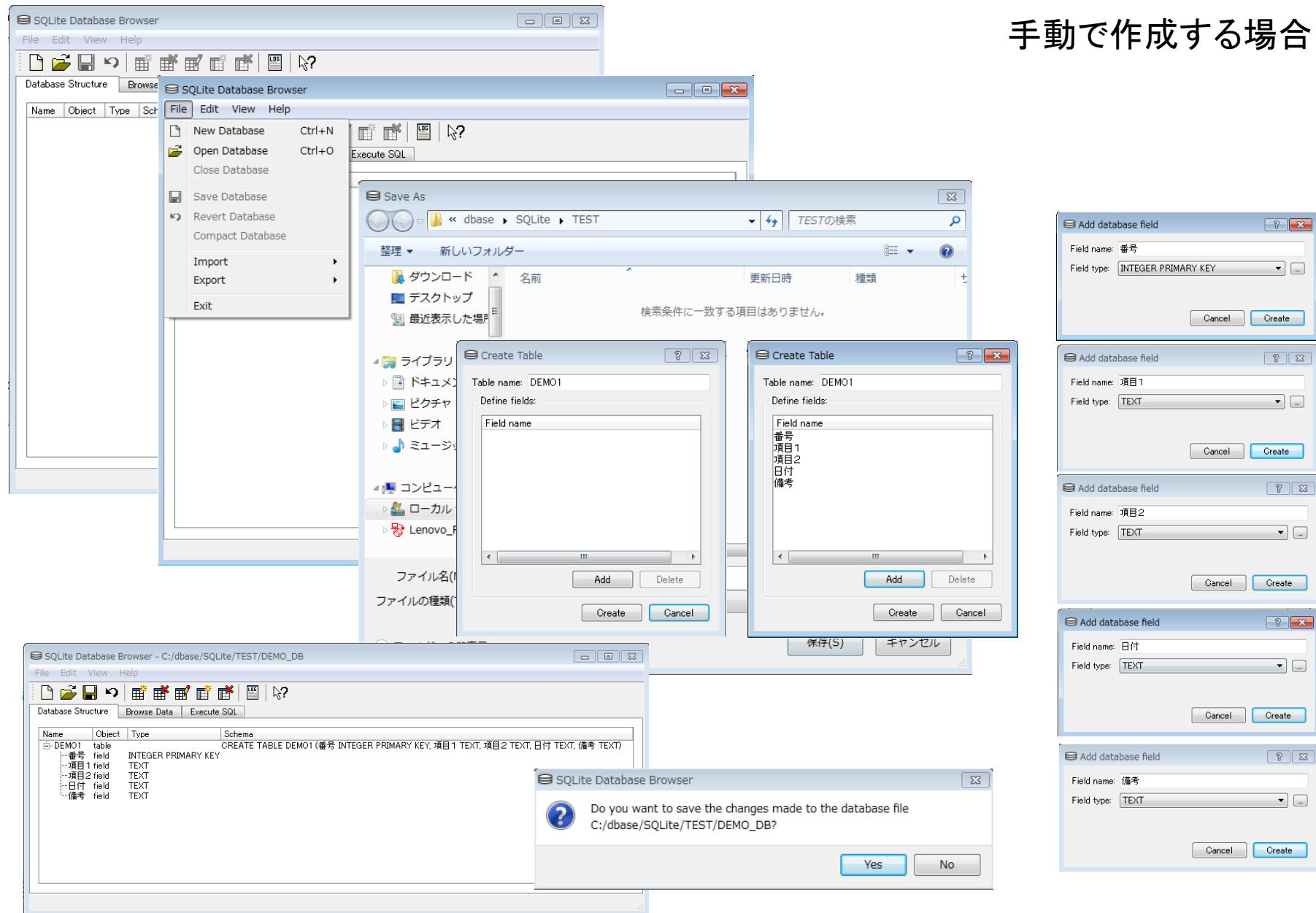


構造が分からなくて、分かっているところから、  
①行目に項目名を決め、  
②行目にサンプルデータを入れていけば良い  
まずは、これだけです！

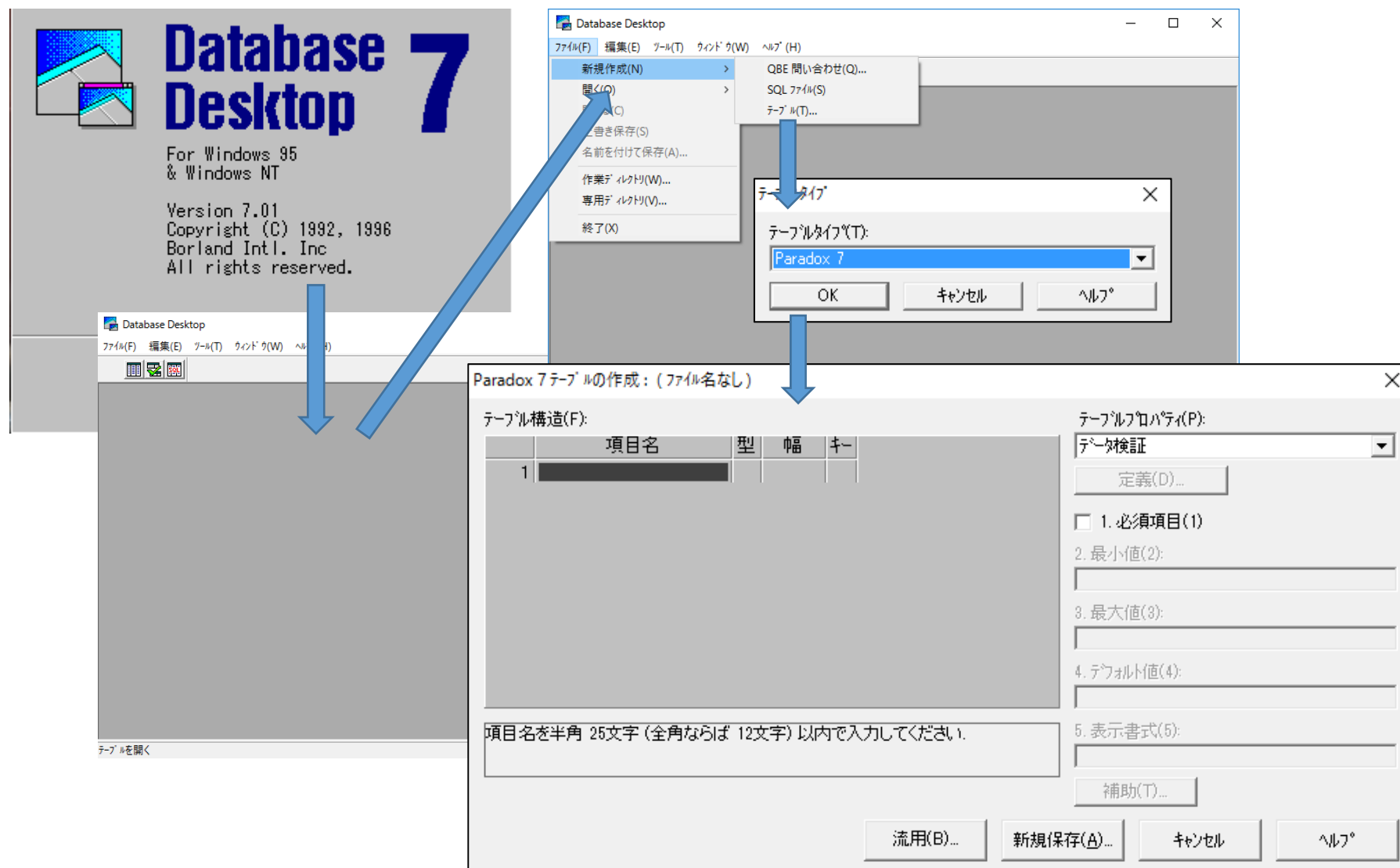
思いつく項目とそのサンプルデータが入力できたら、プログラムが完成した気になってしまいます。

### 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! 以前は項目名が決まったら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へ移行して行きました

### 3) 余談: SQLiteでの困っていたこと

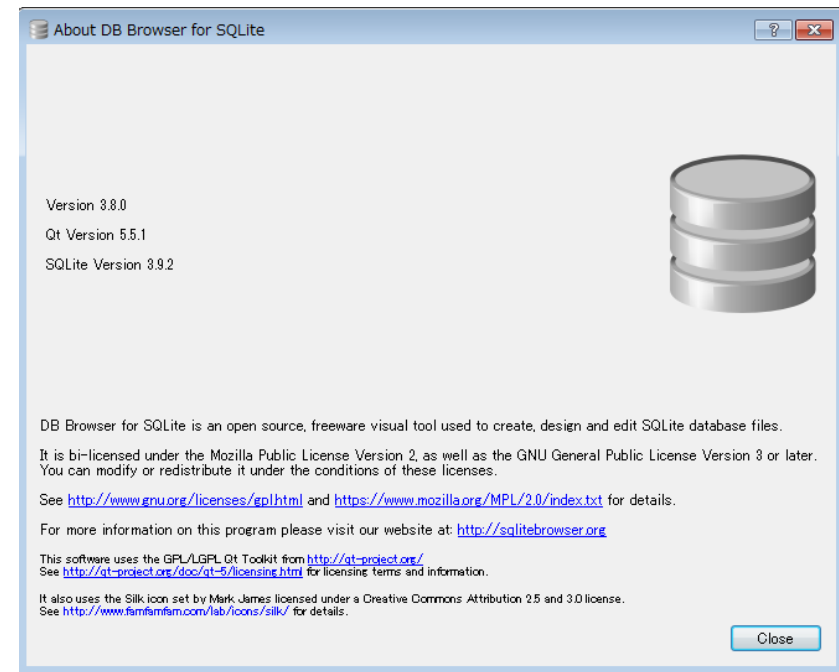
- 1) レコード数か容量がある程度大きくなると上手く動作しない。レコードが表示できなくなる。小規模はOK
- 2) これを書き初めてDB Browser for SQLite 3.8.0のリリースに気が付きました



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

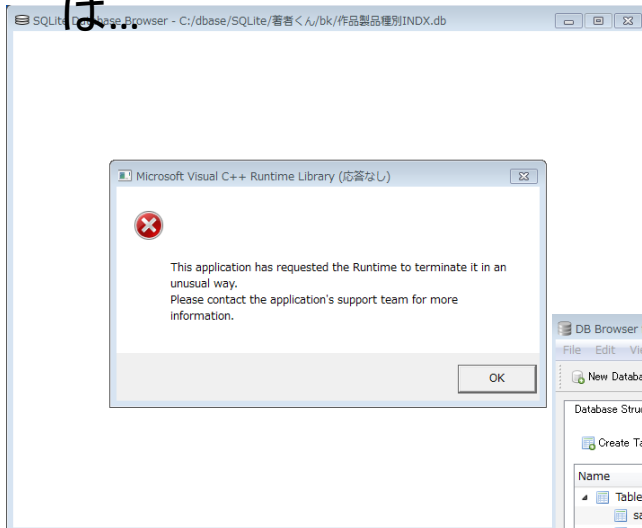
2015/11/13 9:59

Data Base File

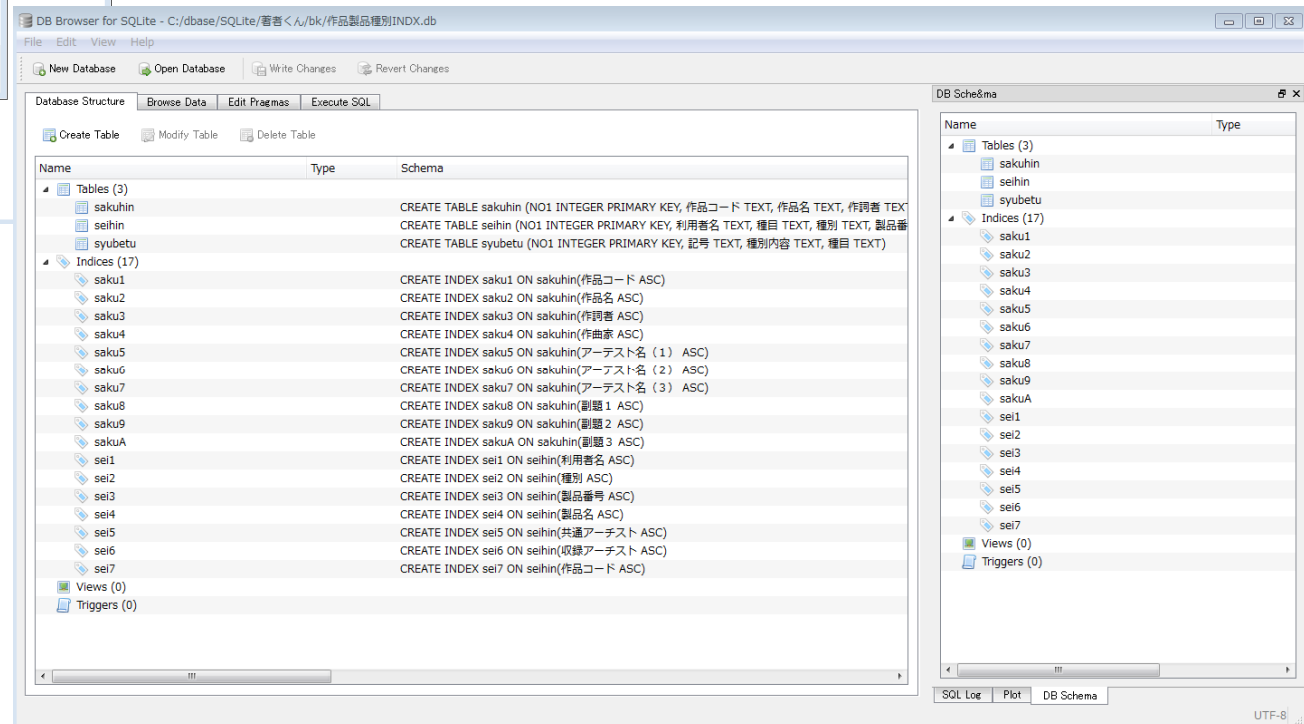
6,645,799 KB

Ver2.0b1 : 例えば6.645GBのサイズのファイルで

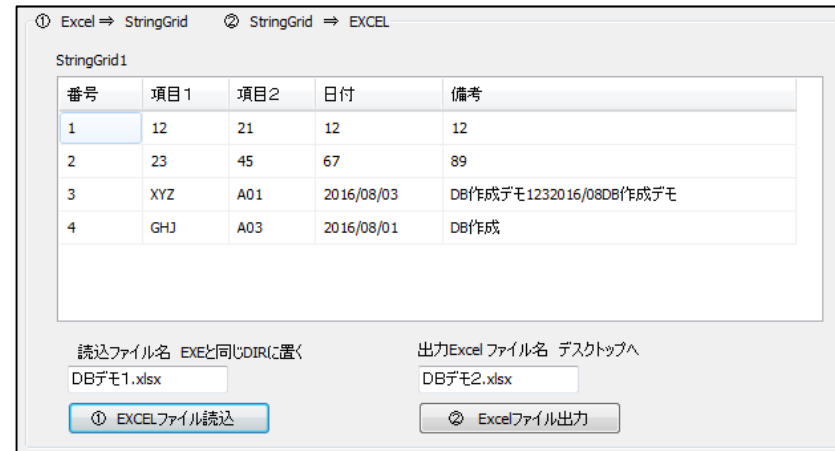
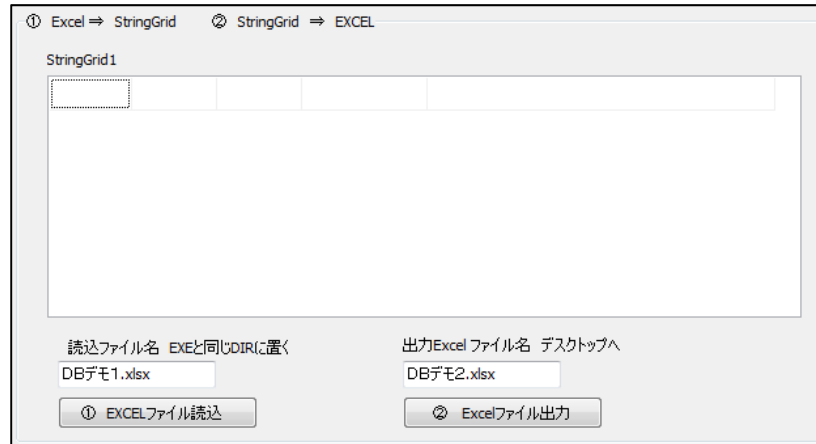
は..



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



- 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10000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:TMyAFrays;  
  //  
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

検索結果表示

検索結果(SG2)

番号	項目1	項目2	日付	備考
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

<SG2>選択行での : ヒット情報

通算文字	Counter	行	項目	番目	内容
23	1	3	4	3	DB作成デモ1232016/08C
42	2	3	4	22	DB作成デモ1232016/08C

END      検索ワード 作成      エラーメッセージ      CLS      Form1.Close

③ レコード検索 レコード内に1つ以上あれば表示

③レコード検索

レコード件数(SG3) 2

Counter	行	項目	番目	内容
1	3	4	3	DB作成デモ1232016/08
2	4	4	3	DB作成

④ ワード検索 全レコード中ヒットものを表示

④検索文字の位置検索

3 ヒット件数(SG4) SG5のヒット位置を表示

Counter	行	項目	番目	内容
1	3	4	3	DB作成デモ1232016/08
2	3	4	22	DB作成デモ1232016/08
3	4	4	3	DB作成

⑤ レコード単位 検索対象文字赤色表示

⑤ 検索ワード表示

Red  Blue  Black       Normal  Bold  Italic  Underline

全レコード

3  
XYZ  
A01  
2016/08/03  
DB作成デモ  
123  
2016/08  
DB作成デモ

選択項目(SG2)の値

DB作成デモ  
123  
2016/08  
DB作成デモ

検索文字位置

④ ヒットしたもの 延全件表示 1レコード(N件⇒N)件表示(SG5)

番号	項目1	項目2	日付	備考
3	XYZ	A01	2016/08/03	DB作成デモ1232016/08DB作成デモ
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

<SG2>選択項目内 のヒット件数 2 選択行 : ヒット情報

通算文字	Counter	行	項目	番目	内容
2	1	3	4	3	DB作成デモ
21	2	3	4	22	DB作成デモ

## 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?)

SQLite Database Browser

Database Structure

Name	Object	Type	Schema
DEMO1	table		CREATE TABLE DEMO1(番号 INTEGER PRIMARY KEY, 項目1 TEXT, 項目2 TEXT, 日付 TEXT, 備考 TEXT)
番号	field	INTEGER PRIMARY KEY	
項目1	field	TEXT	
項目2	field	TEXT	
日付	field	TEXT	
備考	field	TEXT	

SQLite Database Browser

File Edit View Help

New Database Ctrl+N  
Open Database Ctrl+O  
Close Database  
Save Database  
Revert Database  
Compact Database  
Import  
Export  
Exit

Save As

新しいフォルダー

ダウンロード  
デスクトップ  
最近表示した場所

検索条件に一致する項目はありません。

Create Table

Table name: DEMO1

Define fields:

Field name

Add Delete

Create Cancel

Create Table

Table name: DEMO1

Define fields:

Field name

番号  
項目1  
項目2  
日付  
備考

Add Delete

Create Cancel

保存(S) キャンセル

Add database field

Field name: 番号  
Field type: INTEGER PRIMARY KEY

Cancel Create

Add database field

Field name: 項目1  
Field type: TEXT

Cancel Create

Add database field

Field name: 項目2  
Field type: TEXT

Cancel Create

Add database field

Field name: 日付  
Field type: TEXT

Cancel Create

Add database field

Field name: 備考  
Field type: TEXT

Cancel Create

SQLite Database Browser - C:/dbase/SQLite/TEST/DEMO\_DB

File Edit View Help

Database Structure Browse Data Execute SQL

SQLite Database Browser

Do you want to save the changes made to the database file C:/dbase/SQLite/TEST/DEMO\_DB?

Yes No

## 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

The image shows a Delphi IDE window with several components on the top bar: **FDConnection1**, **FDQuery1**, **FDPhysSQLiteDriverLink1**, and **FDGUIxWaitCursor1**. A red bracket groups **FDQuery1**, **FDPhysSQLiteDriverLink1**, and **FDGUIxWaitCursor1**. Below this, a text box says "くどい! :ここをセットしたら自動で...".

The main window is the **FireDAC 接続エディタ - [FDConnection1]**. It has tabs for **定義**, **オプション**, **情報**, and **SQL スクリプト**. The **定義** tab is active, showing a table of parameters:

パラメータ	値
DriverID	SQLite
Pooled	False
Database	.;%base%SQLite%TEST%DEMO_DB.db
User_Name	
Password	
MonitorBy	
OpenMode	CreateUTF8
Encrypt	No
BusyTimeout	10000
CacheSize	10000
SharedCache	True
LockingMode	Normal
Synchronous	Off
JournalMode	Delete
ForeignKeys	On
StringFormat	Choose
GUIDFormat	String
DateTimeFormat	String
Extensions	False
SQLiteAdvanced	
MetaDefCatalog	MAIN
MetaCurCatalog	

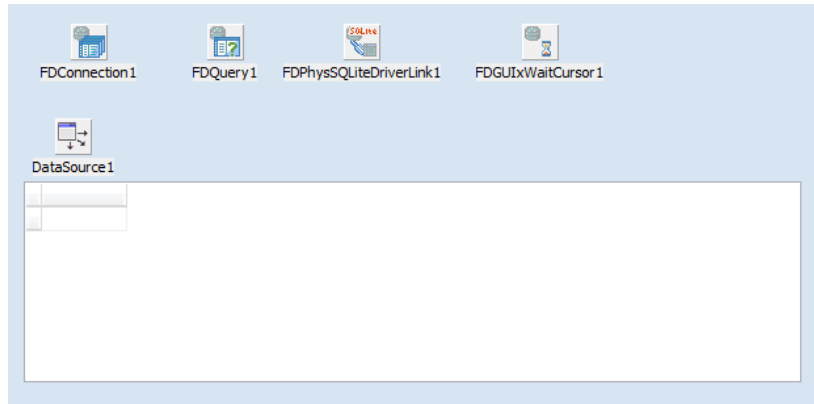
A red box highlights the **DriverID** dropdown menu, which is set to **SQLite**. Another red box highlights the **LockingMode** dropdown menu, which is set to **Normal**. A message box titled **DB\_DEMO - Delphi XE5 - Unit1** is overlaid on the table, displaying the text "接続の確立が成功しました。" (Connection established successfully.) and an **OK** button.

On the right side, the **オブジェクトインスペクタ** (Object Inspector) is shown for **FDQuery1**. The **SQL** property is expanded, showing the text `select * from DEMO1`.

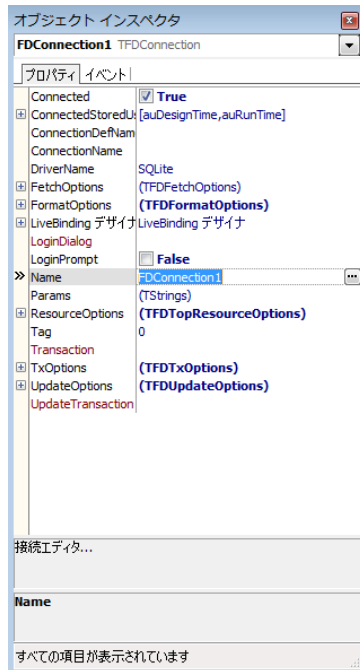
FDQuery1 SQL :select \* from DEMO1

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

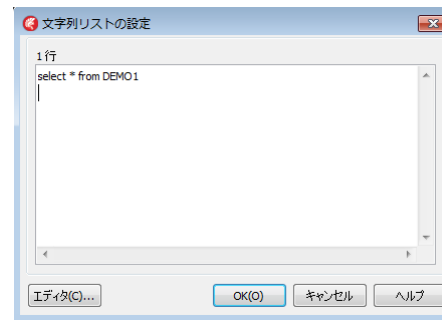
## 4) FireDac (SQLite) 儀式(お経、呪文の類)その2 事前に作ったDBをDataGridで表示させるまで



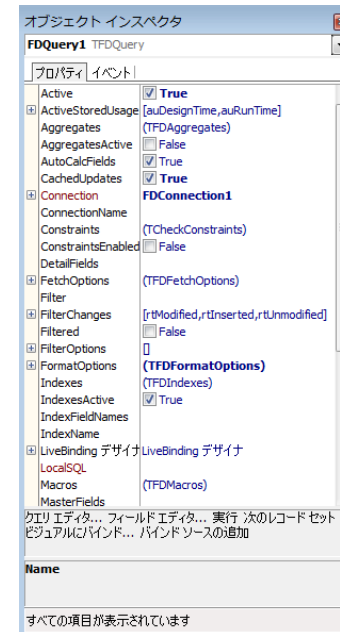
FDConnection1  
Connected :True



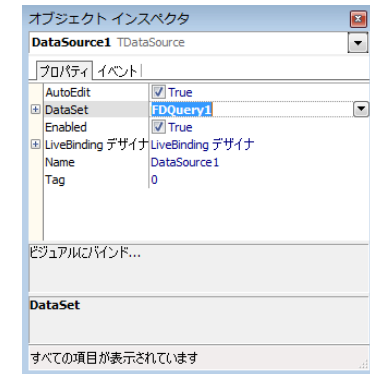
FDQuery1:  
SQL :select \* from DEMO1



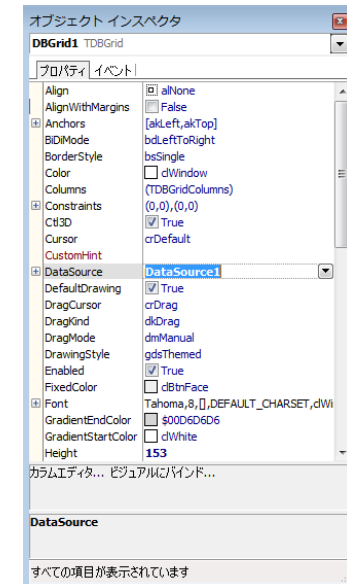
FDQuery1:  
Active:True



DataSource1  
Dataset:FDQuery1

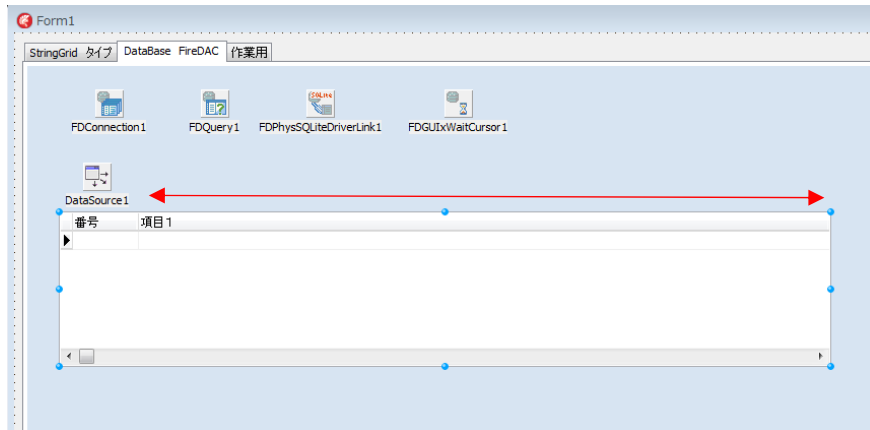


DBGrid1  
Datasource:DataSource1



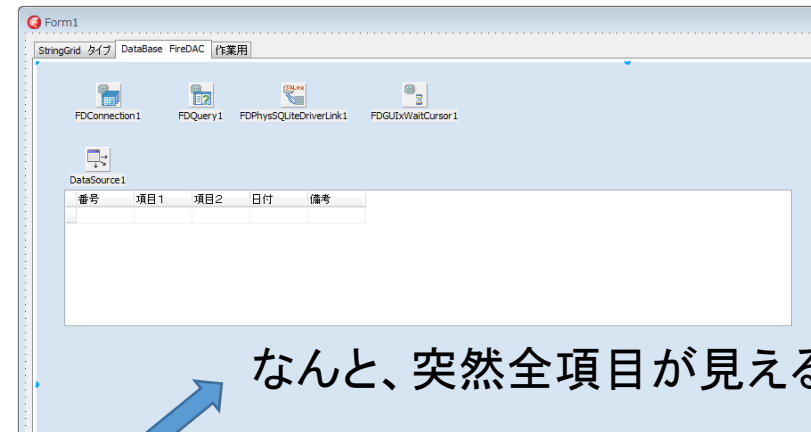
## 4) FireDac (SQLite) 儀式その3

だめだめ問題1: カラムの幅の調整がしにくい

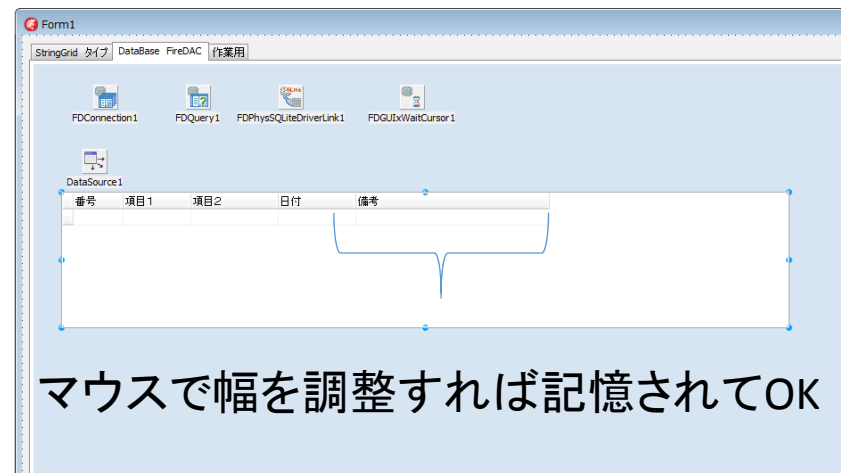
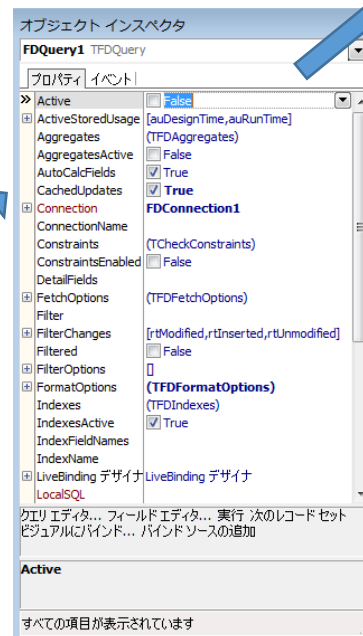
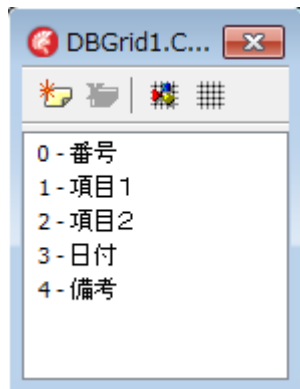


テキスト項目の幅がめちゃ長い！  
調整がしにくい

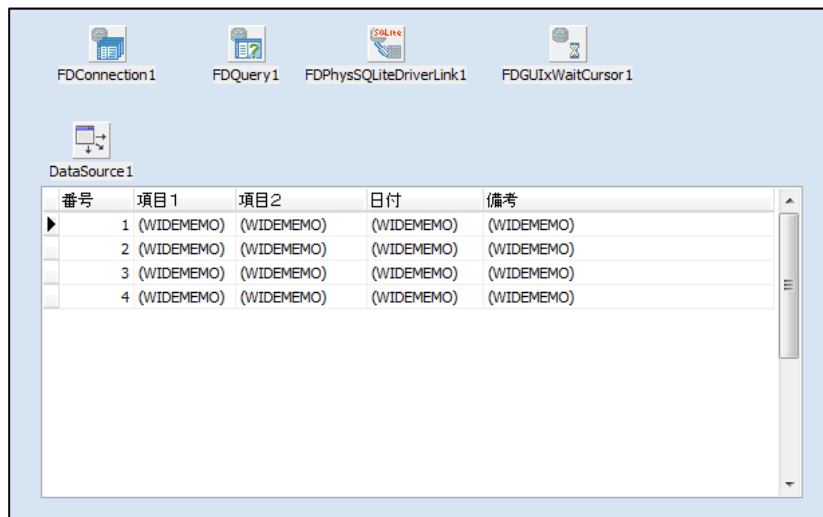
FDQuery1:  
Active:False



カラムエディターを利用



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



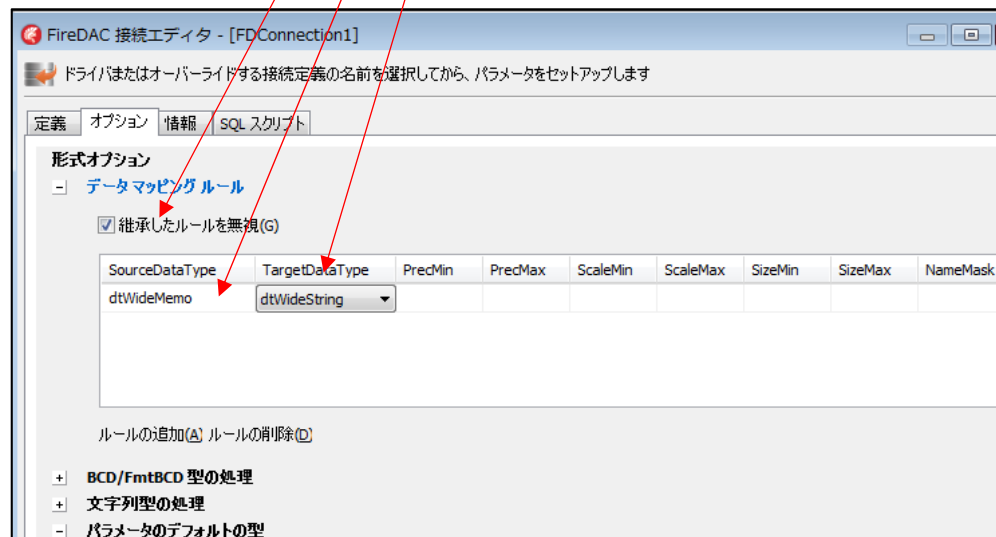
これを解決する方法だけど。。。  
移行についての考え方が間違っていると思う。  
FDDbGridがあるならいいけど  
解決方法が提示されていないのは。。。

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

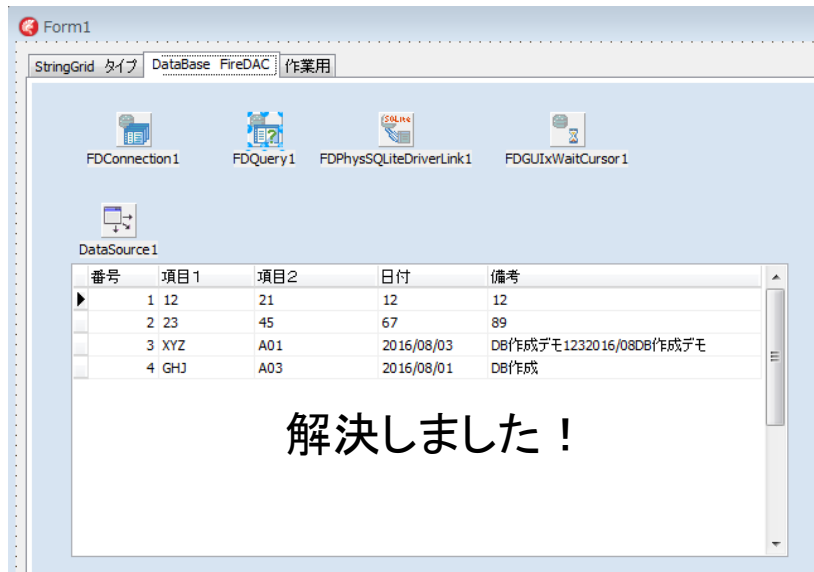
SQLiteでテキストは単に接続すると  
(WIDEMEMO)と表示される

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

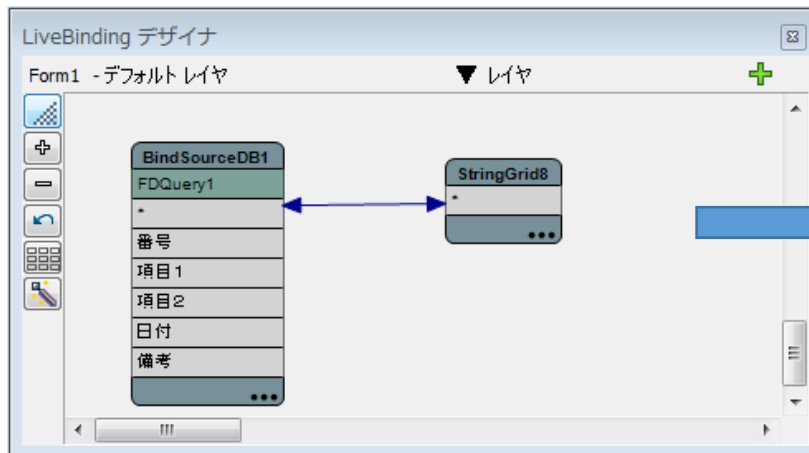


これで行やくDatabaseDesktop  
からFireDac & SQLiteに置き換えられる  
様になりました。

DBGridは(WIDEMEMO)表示のままだが

番号	項目1	項目2	日付	備考
1	(WIDEMEMO)	(WIDEMEMO)	(WIDEMEMO)	(WIDEMEMO)
2	(WIDEMEMO)	(WIDEMEMO)	(WIDEMEMO)	(WIDEMEMO)
3	(WIDEMEMO)	(WIDEMEMO)	(WIDEMEMO)	(WIDEMEMO)
4	(WIDEMEMO)	(WIDEMEMO)	(WIDEMEMO)	(WIDEMEMO)

参考) 以前の方法 ビジュアルバインドを使う

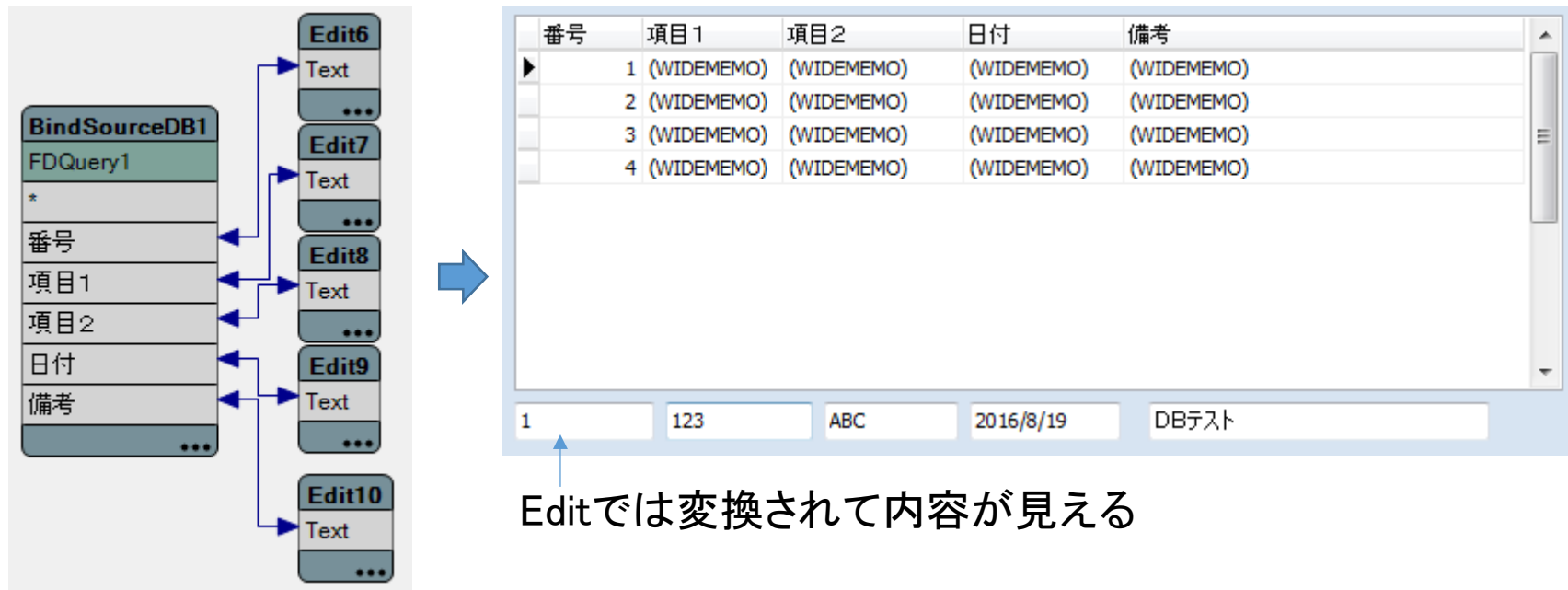


StringGridはノーマルな表示

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

## 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デザイナー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いけない ⇒ めんどくさいし、なによりスマートでない



# 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？とか思っているググ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 TForm1.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;
```



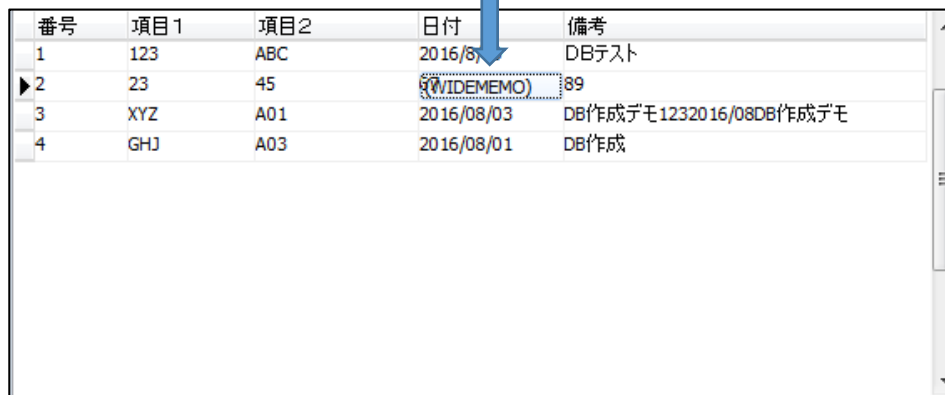
## 4) FireDac (SQLite) ネットでの回避の仕方 ネットの回避の仕方です試してみる OnDrawColumnCellイベントの方法1

- 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

文字が左詰めになって  
しまうが、全部見える  
やった！



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	(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

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す試してみる 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;

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

WidoMemo表示のところだけ  
にすると、表示はOK



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	(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

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

- レコード数が大きい時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

DEMO1テーブル

番号	項目1	項目2	日付	備考

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! ?

StringGrid8 (StringGrid1と同じ)

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1232016/08D
4	GHJ	A03	2016/08/01	DB作成



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1232016/08D
4	GHJ	A03	2016/08/01	DB作成

- 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

## 配列 DML コマンドのパフォーマンス (FireDAC)

表示設定

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

FireDAC には **配列 DML コマンド** をデータベース サーバ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、そのたびクライアントとサーバ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);
```

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 での SQLite の使用

表示設定

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

#### 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。大まかに見れば 5 億回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、Slope、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:** 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 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、ほとんどすべての種類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 サーバ型組み込みクライアント/サーバ

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 高精度の値
5 SQLite の SQL コマンド

## 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`;

The screenshot displays two data grids. The top grid, titled "DEMO1 テーブル", contains the following data:

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

The bottom grid, titled "StringGrid8 (StringGrid1と同じ)", contains the same data:

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

A large blue arrow points upwards from the StringGrid to the table, indicating data flow. At the bottom, there are two navigation buttons: "StringGrid1 ⇒ DEMO1 テーブルへ" and "DEMO1 テーブル ⇒ StringGrid1".

# 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

The image illustrates the steps to create a database file from Delphi:

- FireDAC Connection Editor:** Configure the connection parameters. The **Database** field is set to `C:\dbase\SQLite\TEST\DEMO_DB.db`. The **Test (T)** button is highlighted.
- FireDAC ログイン (Login):** Enter the database name `C:\dbase\SQLite\TEST\DEMO` and click **OK**.
- オブジェクト インспекタ (Object Inspector):** Set the **Connected** property of `FDConnection1` to `True`.
- File Explorer:** Verify that the file `DEMO_DB.db` has been created in the target directory.

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

SQLite Database Browser - C:/dbase/SQLite/TEST/DEMO\_DB.db

File Edit View Help

Database Structure Browse Data Execute SQL

Name	Object	Type	Schema
DEMO1	table		CREATE TABLE DEMO1 (番号 INTEGER PRIMARY KEY, 項目1 TEXT, 項目2 TEXT,...
-番号	field	INTEGER PRIMARY KEY	
-項目1	field	TEXT	
-項目2	field	TEXT	
-日付	field	TEXT	
-備考	field	TEXT	

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

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

DBファイルはFDConnection1で作成

③

StringGrid1 ⇒ テーブル作成用SQL

手で編集後

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

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 StringGrid1 ⇒ DEMO1テーブルへ

```
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

⑥ DEMO1テーブル ⇒ StringGrid1

```
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

⑦ Excelファイル出力

```
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

編集

番号	項目1	項目2	日付
4	GHJ	A03	2016/08/01

備考  
DB作成

DEMO1更新

1

DEMO1 項目名取得

番号  
項目1  
項目2  
日付  
備考

4

Record Count

DEMO1 番号 ASC  
DEMO1 番号 DESC  
DEMO1 1レコード追加  
FindFirst  
FindLast  
Prior  
Next  
Move to  
DEMO1 オープン  
DEMO1 クローズ

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！。

SQL検索

検索文字 対象項目 検索モード 表示順序

検索    全項目  あいまい  一致  NULL  昇順  降降

AND    全項目  あいまい  一致  NULL

AND    全項目  あいまい  一致  NULL

SQL= select \* from DEMO1

# 5) FileMakerProでレポート作成 ①形式変換

The first screenshot shows the FileMaker Pro 10 Advanced 'QuickStart' dialog. Under 'データベースの作成' (Create Database), the 'Starter Solution' option is selected. A red box highlights the 'データベースを開く' (Open Database) icon. The second screenshot shows the '新規作成するファイルの名前' (New File Name) dialog. The file name is 'DBデモ2 (変換).fp7' and the type is 'FileMaker ファイル (\*.fp7)'. A red arrow points from the '開く(O)' button in the first screenshot to the '開く(O)' button in this dialog. The third screenshot shows the 'DBデモ2 (変換)' database window. It displays a table with 4 records and 5 columns: 番号, 項目1, 項目2, 日付, and 備考.

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	GHI	A03	2016/08/01	DB作成

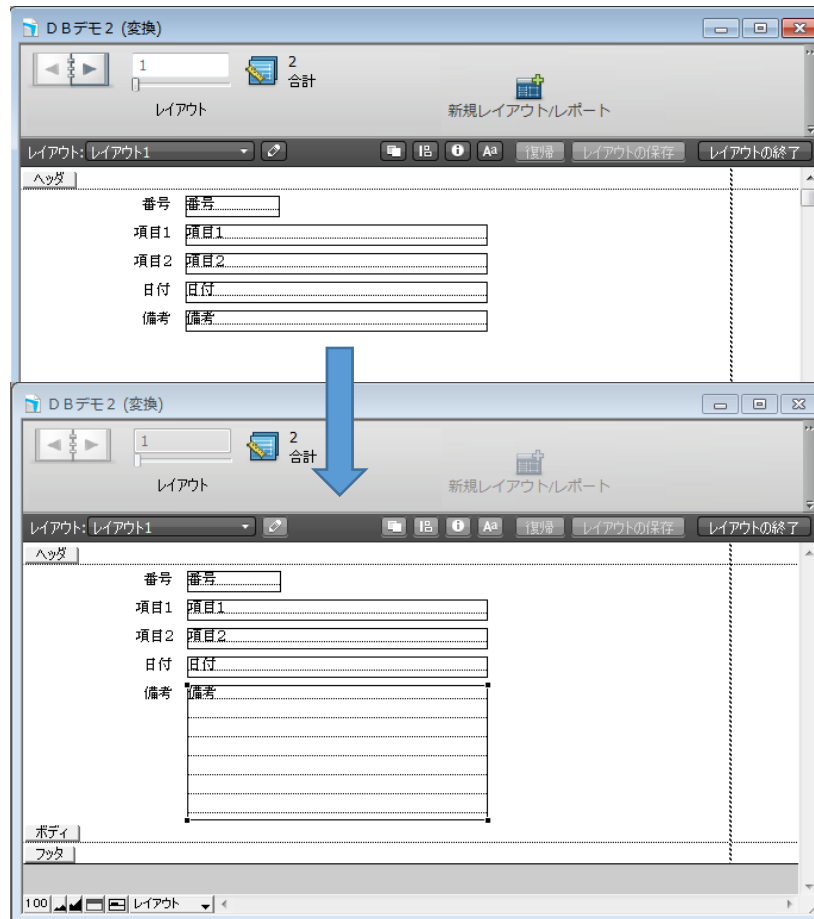
FileMakerProはレポートデザインをしながらDB項目を作れるので。。。!  
FistRepotもあるが...

堅苦しいことは言わずにもうファイルメーカーとして作成される

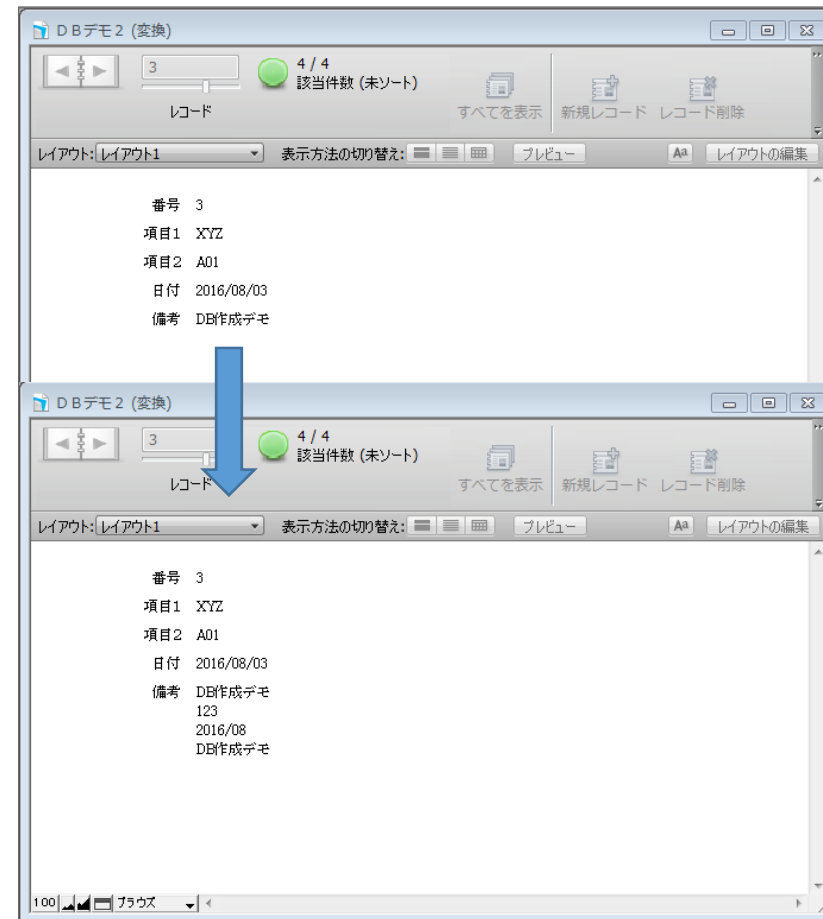
# 5) FileMakerProでレポート作成 ②レポート形式

レイアウト1:レポート形式 デフォルト

レイアウト編集



ブラウザ



# 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



ブラウザ画面

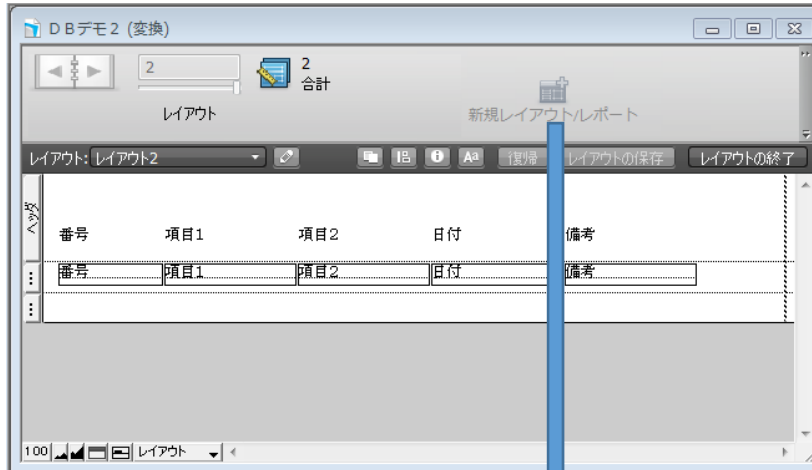


# 5) FileMakerProでレポート作成 ②表形式

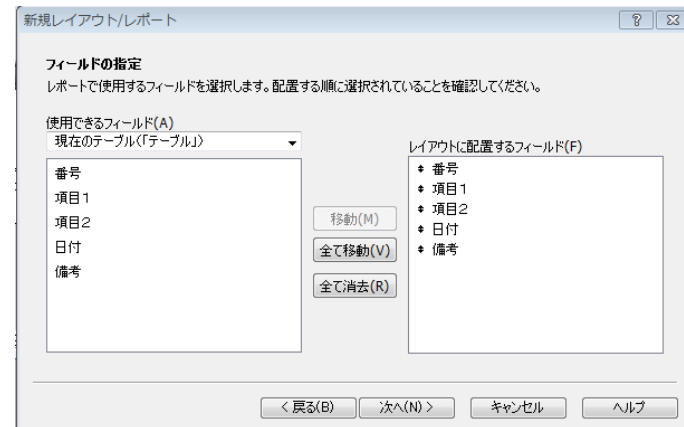
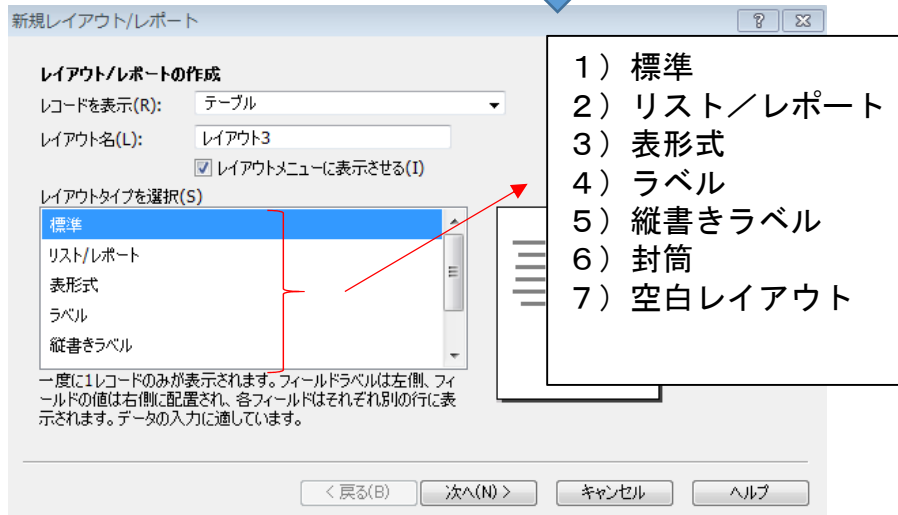
レイアウト2:表形式:デフォルト

レイアウト編集画面

ブラウザ画面



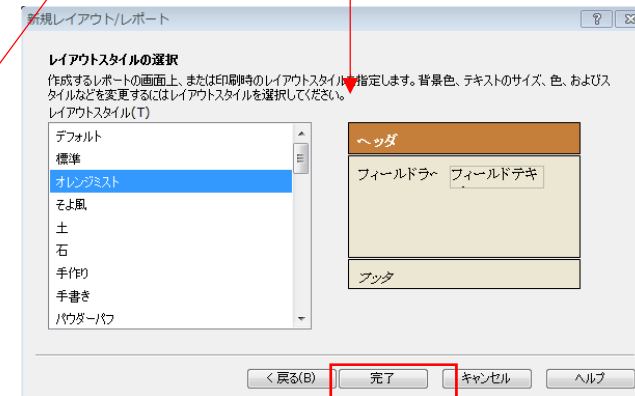
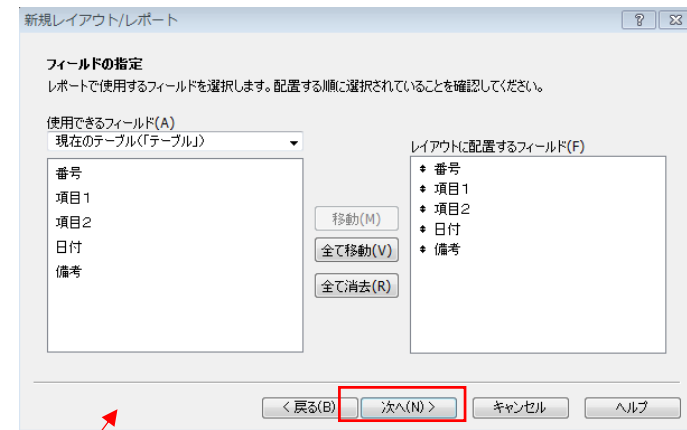
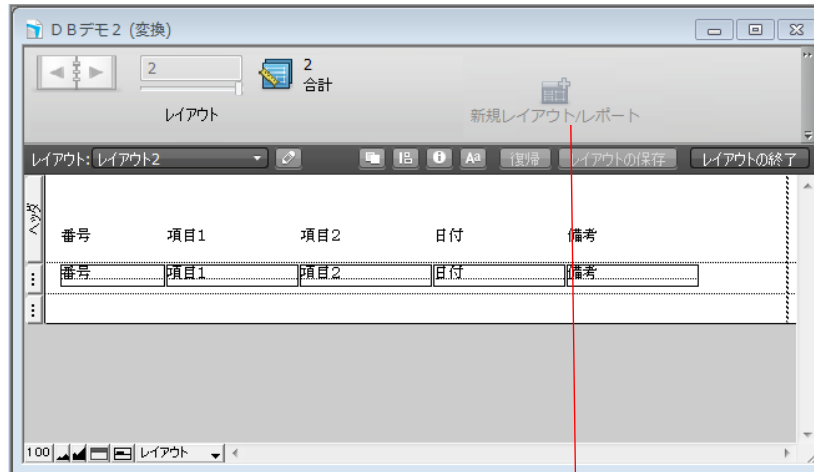
レイアウトの追加



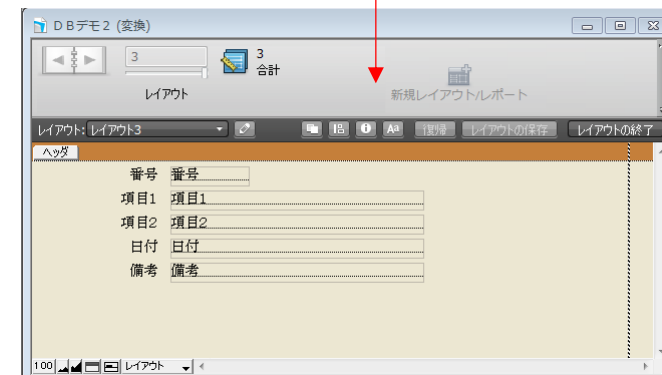
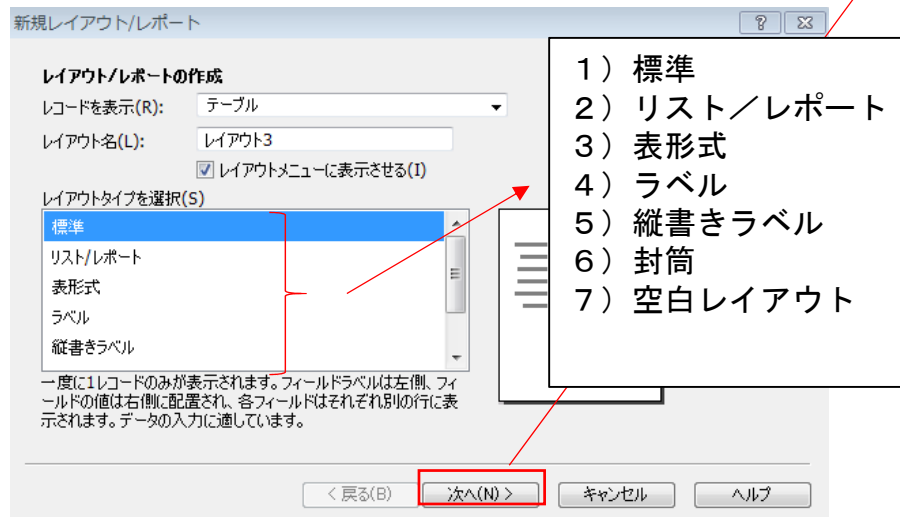


# 5) FileMakerProでレポート作成 ③レイアウト追加

## レイアウト編集



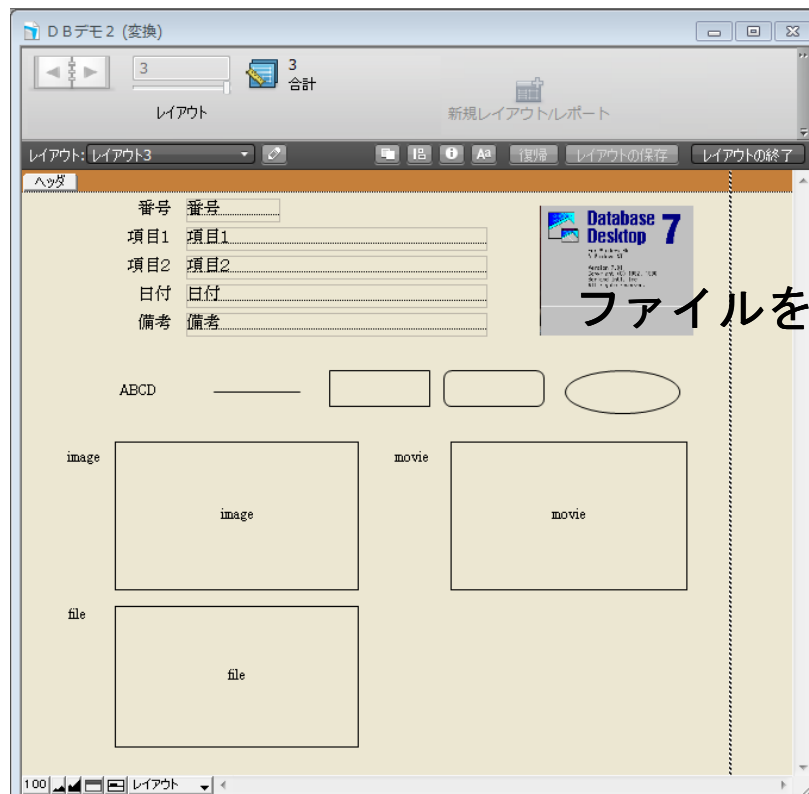
## レイアウトの追加



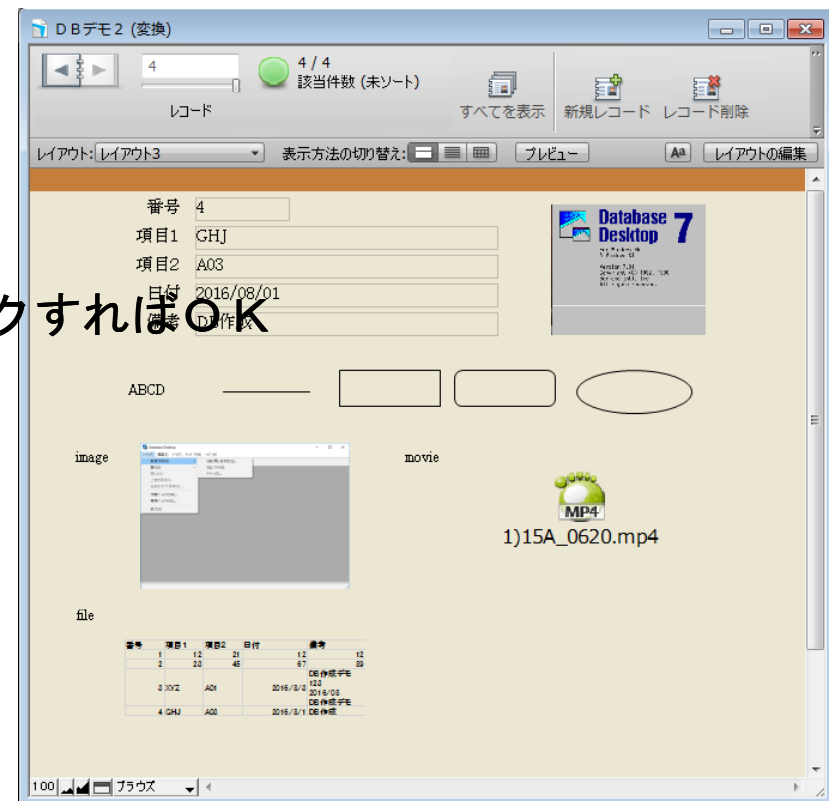
# 5) FileMakerProでレポート作成 ④レイアウト編集

ちょっとしたドロワー機能  
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) ファイルメーカー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あ！  
これからはC++BuilderStarterか！ DelphiStarterか！

