CHAPTER 1

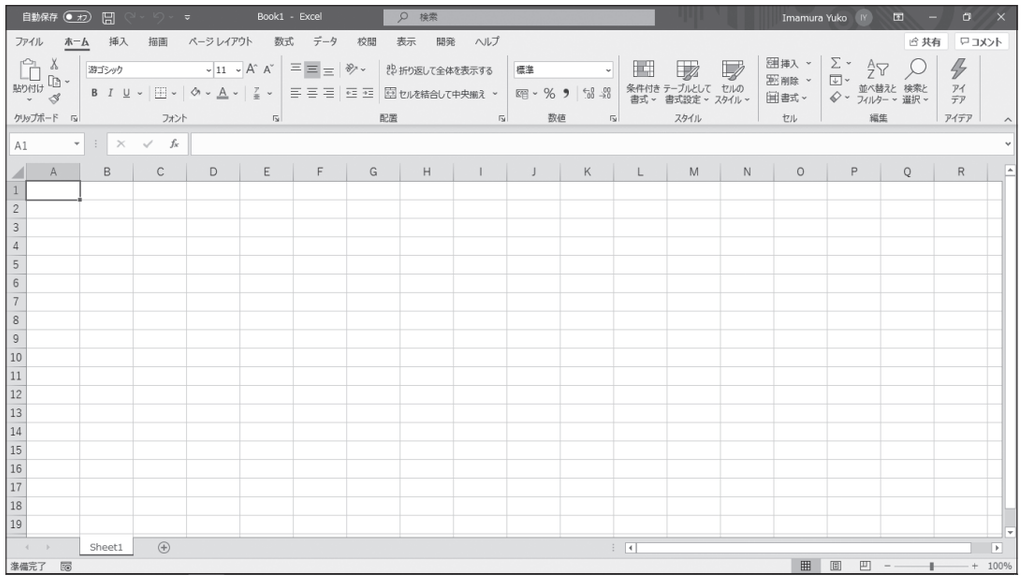
1-1 Excelの得意・不得意

Excelは多機能な表計算ソフトですが、なんでもかんでもExcelで処理するのは、効率的とはいえません。まずはExcelというソフトの得意分野と不得意分野について理解しましょう。

1-1-1 Excelは「なんでもできてしまいそう」

Excel（**図1**）は、Microsoft社製の大変有名なソフトウェアです。PCの分野では標準の表計算ソフトとして扱われ、会社の規模を問わず、ビジネスシーンで多くの人に使われています。数値データを元にさまざまな数式を組み込んで自動で計算をしたり、美しい集計表やグラフなどを素早く作成したりと、その用途は多岐にわたります。

図1 Excel



しかし、できることが多すぎて、「なんでもExcelで作ってしまう」というケースもしばしば見られます。日報やレポート、ときにはセルを正方形にして（Excel方眼紙と呼ばれる）あらゆる形態の文書をExcelで作成する、という人もいます。もちろんExcel は1つのツールですので、利用者が便利だと思う使い方をするのに問題はありません。しかし、自由に使いすぎて本来の優れた表計算機能が十分に発揮できなくなってしまい、「効率的でない」場合があります。

なんでも作れてしまいそうなExcelですが、「表計算ソフト」である以上、得意・不得意がちゃんとあります。本書ではExcelの特性を理解し、もう一歩進んだ使い方を学んでいきます。

1-1-2 Excelが実は苦手なこと

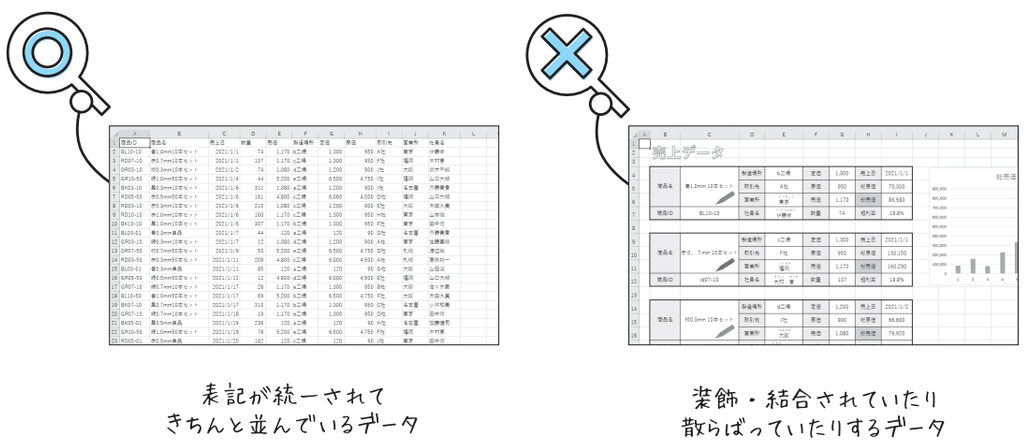
Excel には不得意分野があります。まず1つは、「膨大な量」のデータを貯め込むことはそれほど得意ではありません。

Excel 2007以降のシートは、行（縦方向）は1,048,576、列（横方向）は16,384が上限ですが、ここまでいかなくても行と列を多く使いすぎると、動作が遅くなったり、フリーズしてしまったりと、不具合が起きやすくなってしまいます。そもそも、多くのデータを詰め込んでいたり、数多くのシートを使っていたりすると、可読性も効率も悪くなってしまいます。また、あちこちにデータが散らばってしまうと、必要なときにデータを「探す」という無駄な作業が発生してしまいます。

もう1つ、Excelは**加工しやすいデータ**を維持・管理することが苦手です。

加工しやすいデータとは、一定のルールに則って、**整列・整頓されたデータ**です。同じ種類のデータは同じ列に並べる、要素の表記を統一する（半角や全角、大文字や小文字などの表記がバラバラだと、違う要素だと判断される）、などのルールがきちんと守られていないと、集計や分析を行う効率は大幅に下がり、Excelの機能を有効に使うことができません（**図2**）。

図2 加工しやすいデータとは



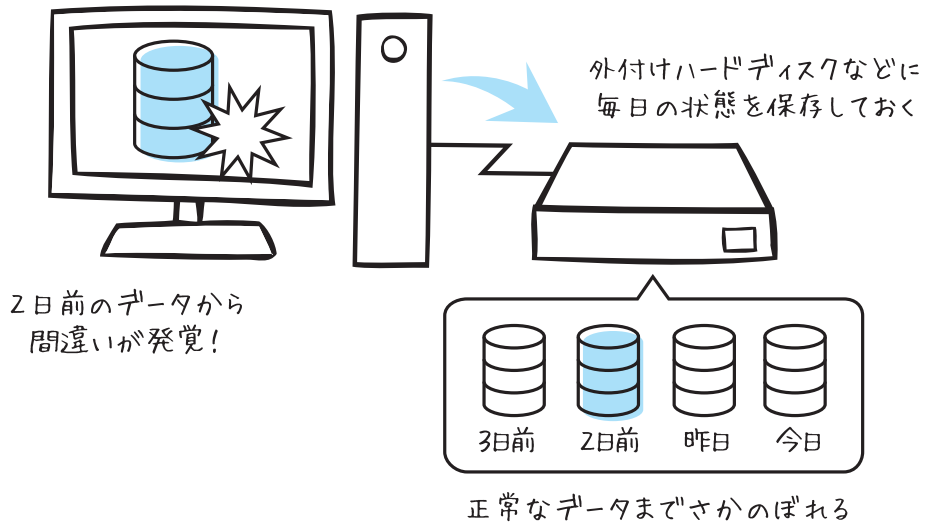
1-4-4 バックアップ

データベースは、さまざまなデータ利用のための情報源であり、組織の重要な財産です。データをExcelなどに取り出していくら手を加えても元データには影響を及ぼさないので、データの保全性が高まり、検索・集計なども効率的になるので、正しく使うことができれば生産性は大幅に向上することでしょう。

ただし、信頼性の高い情報源であるということは、替えが利かないということでもあります。財産であるデータベースが破損、消失してしまったら大変な損害です。そのデータベースが保存されているパソコン自体の破損も十分考えられます。その対策のために、使用するパソコンとは別のハードディスクなどに、定期的にファイルの複製を作っておきましょう。これを**バックアップ**と呼びます。

頻繁に更新されるデータベースならば、バックアップツールを利用して、自動的に毎日決まった時間に外部へバックアップを作成することをおすすめします。また、バックアップファイルは数日から数週間分残しておくと、ファイルが破損した時だけでなく、間違いが発覚したときなどにどこまでが正しいデータなのかを特定して、そのファイルへ戻ることができます（**図14**）。

図14 データベースファイルは必ずバックアップ



CHAPTER 2

2-3 書込に理想的なExcelのシート

Excelのデータを Accessに書き込む際、普段使っているExcelのシートそのままの状態では、うまくいかない場合がほとんどです。どのような状態のシートであればスムーズに書き込むことができるのでしょうか。

2-3-1 利用・加工に適したデータとは

Accessには、**表形式に整理整頓された純粋なデータ**しか読み込むことができません。

ここでいう「整理」とは不要なものを削除すること、「整頓」は使いやすい形に変更して再配置すること、という意味です。

Excelのシートを Accessに読み込ませた場合、反映できるものを**表2**にまとめました。

表2 Excelから Accessへの読込の可否

要素	結果
値	可
数式	数式そのものではなく、式の結果が値として格納
結合セルの値	結合範囲の先頭のセルのみ格納され、他は空白となる
フォントや色などの書式	不可
セルのコメント	不可
ふりがな	不可
ワードアート	不可
画像・グラフ	不可
テキストボックスなどのオブジェクト	不可

こうして見ると、Excelで主に使われる機能はほとんど Accessへは反映できないことがわかりますね。書き込めないもの、不要なものなどはあらかじめExcelから削除しておかねばなりません。

2-3-2 Excelデータはそのまま使えない場合が多い

たとえば、ボールペンの売上データを記録した**図11**のようなExcelシートがあるとします。

図11 普段使っているシートの例

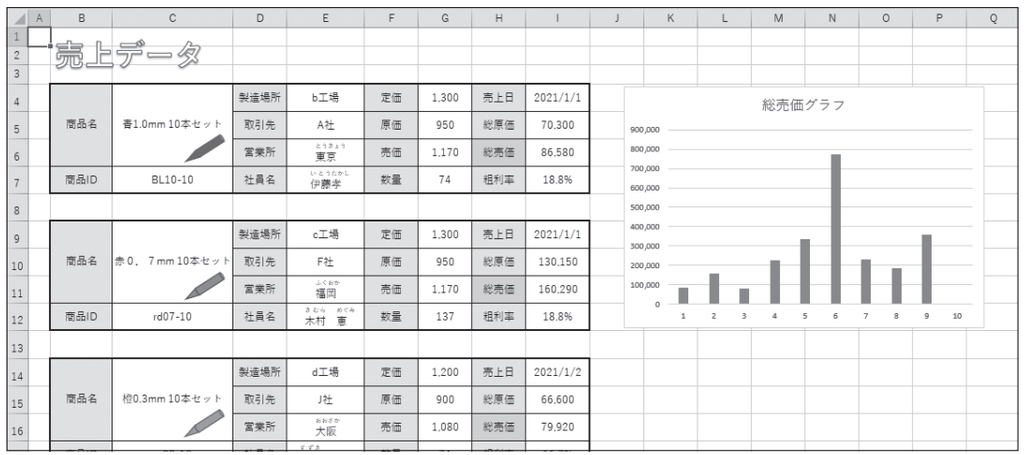


表2から判断して、かなりの部分がAccessに書き込むことができません。このExcelのデータを Accessに書き込むことができるように修正すると、**図12**のようになります。

図12 Accessに書き込むことができるシートの例

商品ID	商品名	売上日	数量	売価	製造場所	定価	原価	取引先	営業所	社員名
BL10-10	青1.0mm10本セット	2021/1/1	74	1,170	b工場	1,300	950	A社	東京	伊藤孝
RD07-10	赤0.7mm10本セット	2021/1/1	137	1,170	c工場	1,300	950	F社	福岡	木村恵
OR03-10	橙0.3mm10本セット	2021/1/2	74	1,080	d工場	1,200	900	J社	大阪	鈴木千絵
GR10-50	緑1.0mm50本セット	2021/1/4	44	5,200	e工場	6,500	4,750	I社	福岡	山口大輔
BK03-10	黒0.3mm10本セット	2021/1/6	311	1,080	a工場	1,200	900	I社	名古屋	斉藤美香
RD05-50	赤0.5mm50本セット	2021/1/6	161	4,800	c工場	6,000	4,500	D社	福岡	山口大輔
RD03-10	赤0.3mm10本セット	2021/1/6	215	1,080	c工場	1,200	900	E社	大阪	吉田久美
RD10-10	赤1.0mm10本セット	2021/1/6	160	1,170	c工場	1,300	950	H社	東京	山本徹
BK10-10	黒1.0mm10本セット	2021/1/6	307	1,170	b工場	1,300	950	F社	東京	田中博
BL03-01	青0.3mm単品	2021/1/7	44	120	a工場	120	90	D社	名古屋	斉藤美香
GR03-10	緑0.3mm10本セット	2021/1/7	12	1,080	d工場	1,200	900	A社	東京	佐藤美穂
OR07-50	橙0.7mm50本セット	2021/1/9	53	5,200	e工場	6,500	4,750	D社	札幌	渡辺綾
RD03-50	赤0.3mm50本セット	2021/1/11	209	4,800	c工場	6,000	4,500	A社	札幌	高橋裕一
BL03-01	青0.3mm単品	2021/1/11	85	120	a工場	120	90	G社	大阪	山田誠
GR05-50	緑0.5mm50本セット	2021/1/12	12	4,800	d工場	6,000	4,500	E社	福岡	山口大輔
GR07-10	緑0.7mm10本セット	2021/1/17	28	1,170	e工場	1,300	950	B社	大阪	佐々木昇
BL10-50	青1.0mm50本セット	2021/1/17	69	5,200	b工場	6,500	4,750	F社	大阪	吉田久美
BK07-10	黒0.7mm10本セット	2021/1/17	310	1,170	b工場	1,300	950	C社	名古屋	小林和美
GR07-10	緑0.7mm10本セット	2021/1/18	19	1,170	e工場	1,300	950	D社	東京	田中博
BK05-01	黒0.5mm単品	2021/1/19	236	120	a工場	120	90	H社	名古屋	加藤健司
GR10-50	緑1.0mm50本セット	2021/1/19	78	5,200	e工場	6,500	4,750	F社	福岡	木村恵
RD05-01	赤0.5mm単品	2021/1/20	162	120	c工場	120	90	J社	東京	田中博

CHAPTER 3

3-2 VBAの基礎知識

Accessでテーブルの準備ができたら、次はExcelでVBAを使う準備をします。CHAPTER 3では、VBAを扱うための基礎的なことを学んでいきましょう。

3-2-1 xlsm形式（マクロを含む形式）で保存する

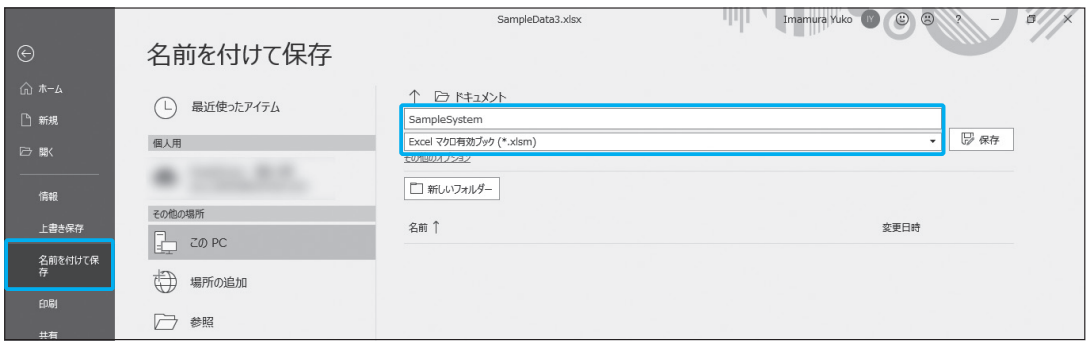
Excelは、マクロを含まないファイルは「.xlsx」、マクロを含むファイルは「.xlsm」という拡張子で区別されています。VBAを使うということは、マクロを含むファイルにしなければならないので、「.xlsm」でファイルを作成し直します。

サンプルファイルの「CHAPTER3」フォルダー→「Before」フォルダーに収録されているSampleData3.xlsxを開いてください。このファイルはCHAPTER 2で処理したものと同様に整理整頓・表記の統合がなされています。

このファイルには、Accessで主キーとして設定した「販売ID」がありませんが、「販売ID」のデータ型を**オートナンバー型**（他のフィールドを書き込む際に自動で数値が振られる型）に設定したので、Excel側では「販売ID」の追加は不要です。

では、拡張子をxlsmに変更しましょう。リボンの「ファイル」タブをクリックし、「名前を付けて保存」を選択します。保存場所を指定して、ファイルの種類を「Excelマクロ有効ブック（*.xlsm）」にして保存します。ファイル名は「SampleSystem.xlsm」とします（**図7**）。

図7 「.xlsm」拡張子で保存



CHAPTER 3

3-3 VBAコード内でADOとSQLを利用する

それでは、ここからはデータベースと連携させた実用的なコードを記述していきましょう。

3-3-1 全体の流れとINSERT構文

MessageManagerモジュールに記述してきた3-2までのコードをすべて削除して、**コード4**を書きます。ちょっと長いですが、後でブロックに分けて解説するので、まずは書いてみてください。P.89までの変更がなされたサンプルが「CHAPTER3」フォルダー→「Before」フォルダーにSampleSystem3.xlsmとして収録されていますので、そちらを利用しても構いません。

3-2-3では「挿入」→「プロシージャ」で自動挿入しましたが、必要な記述がなされていれば直接入力でもプロシージャと認識されます。

なお、「'（シングルクォーテーション）」より右側は**コメントアウト**と呼び、実行されない領域になります。後からプログラムを読み返したときに内容がわかるように、プログラム中に作成者がメモを残すイメージです。なお、本書ではこれ以降、モジュールの先頭に「' # モジュールの概要」、プロシージャの先頭に「' ## プロシージャの概要」と、適宜解説のためのコメントを入れてプログラムを記述していきます。

コード4 Accessへレコードを挿入するためのコード

```
01 '# シート上データの移行  ←モジュールの概要
02 Option Explicit ←変数の宣言を強制する
03
04 Public Sub insertRecord()
05   '## レコードの挿入  ←プロシージャの概要
06
07   '接続
08   Dim cn As Object ←ADOコネクション用オブジェクトの宣言
09   Set cn = CreateObject("ADODB.Connection") ←ADOコネクションを作成
10   cn.Open "Provider=Microsoft.ACE.OLEDB.12.0;" & _
11           "Data Source=" & ThisWorkbook.Path & "¥Data.accdb;" ←
12                                                 Accessファイルを指定してコネクションを開く
13   'SQL文の作成
14   Dim sql As String ←SQL文用変数の宣言
15   sql = _
16     "INSERT INTO 販売管理(" & _
17       "商品ID, " & _
18       "商品名, " & _
19       "売上日, " & _
20       "数量, " & _
21       "売価, " & _
22       "製造場所, " & _
23       "定価, " & _
24       "原価, " & _
25       "取引先, " & _
26       "営業所, " & _
27       "社員名) " & _
28     "VALUES(" & _
29       "'BL10-10', " & _
30       "'青1.0mm10本セット', " & _
31       "#2021/1/1#, " & _
32       74 & ", " & _
33       1170 & ", " & _
34       "'b工場', " & _
35       1300 & ", " & _
36       950 & ", " & _
37       "'A社', " & _
38       "'東京', " & _
39       "'伊藤孝');"
40
41   '実行
42   cn.Execute sql
43
44   '接続解除
45   cn.Close
46   Set cn = Nothing
47
48 End Sub
```

このモジュールはExcel VBAを使ってAccessへレコードを移行する目的で使うので、モジュール名を「M_DataTransfer」へ変更します。ここより、「モジュール」や「シート」など多岐に渡るオブジェクトをコードで指定しなければならないので、先頭に種類の頭文字を付けることとします（シートのオブジェクト名も「S_Sales」になっています）。

ここまで変更を加えたものが、**図24**です。

CHAPTER 3

3-5 関数の戻り値によって動作を変える

エラーメッセージだけでなく、正常終了時にもメッセージがあった方が親切です。また、誤動作を防ぐために処理前に確認メッセージを出す機能も追加してみましょう。

3-5-1 処理前メッセージ

3-4-2で、指定したシートの最終行を取得するgetLastRowという関数を作りました。このように関数というのは引数として値を渡し、それに対応した値を受け取る（**戻り値**と呼びます）ことができるのですが、実はここまでなんどか書いてきたMsgBoxも関数なのです。

コード15（24行目）のエラーメッセージを出力する部分でMsgBoxを使っていますが、詳しく見ると**図42**のようになっており、ボタンの種類は**表4**、アイコンスタイルは**表5**のように表記します。

図42 MsgBox

```
MsgBox “テキスト”, ボタンの種類 + アイコンスタイル , “タイトル”
```

ボックス内に表示されるテキスト

タイトルバーに表示されるテキスト（省略可）

表4 ボタンの種類

定数	値	表示されるボタン
vbOKOnly	0	「OK」のみ（規定値/省略可）
vbOKCancel	1	「OK」「キャンセル」
vbAbortRetryIgnore	2	「中止」「再試行」「無視」
vbYesNoCancel	3	「はい」「いいえ」「キャンセル」
vbYesNo	4	「はい」「いいえ」
vbRetryCancel	5	「再試行」「キャンセル」

CHAPTER 4

4-5 クエリの種類とその他の活用方法

ここまでの解説でいくつかのクエリを使ってデータを操作してきましたが、クエリにはさまざまな種類と、便利な使い方があります。ここではその例をかんたんに紹介します。

4-5-1 クエリの種類

Access のクエリは大きく分けて、データを計算したり抽出したりする**選択クエリ**（表5）と、テーブル内のデータに変更を加える**アクションクエリ**（表6）に分類することができます。**2-6** で扱ったのはアクションクエリです。

表5 選択クエリの例

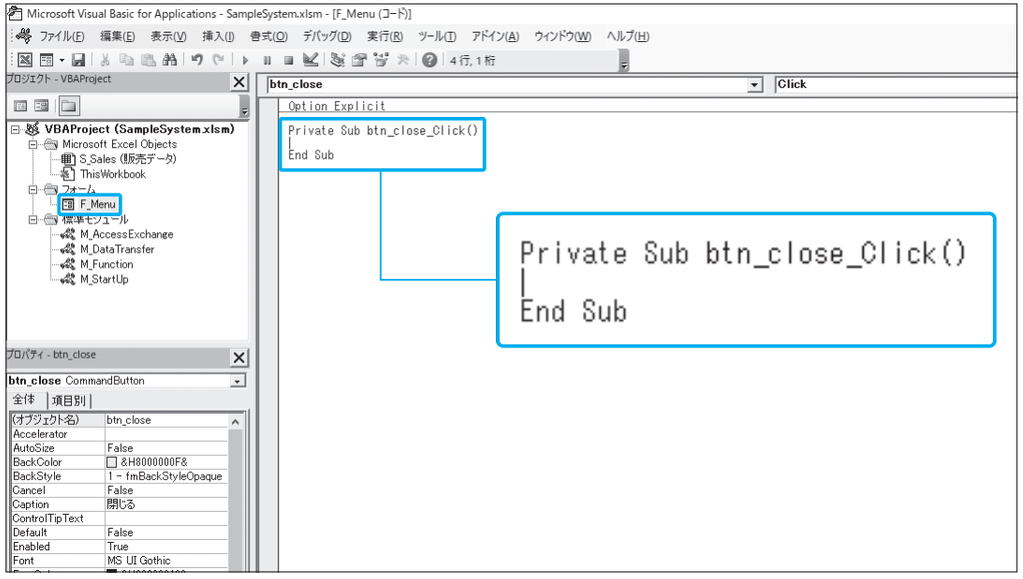
種類	内容
選択クエリ	条件を付けてデータの抽出や並べ替えを行う
パラメータクエリ	条件の値をユーザーへ入力要求してクエリを実行する
重複クエリ	テーブル内で重複したレコードを抽出する
不一致クエリ	2つのテーブルの不一致レコードを抽出する
集計クエリ	データをグループ化して集計する
クロス集計クエリ	行列の項目が交差する位置で集計する

表6 アクションクエリの例

種類	内容
更新クエリ	テーブルのデータを更新する
追加クエリ	テーブルにレコードを追加する
削除クエリ	テーブルのレコードを削除する

本書のサンプルのような売上データをグループごとに合計するときなどは**集計クエリ**が非常に有用です。例として**クロス集計クエリ**を使ってアイテムごとの月別の合計売上数を算出してみましょう。

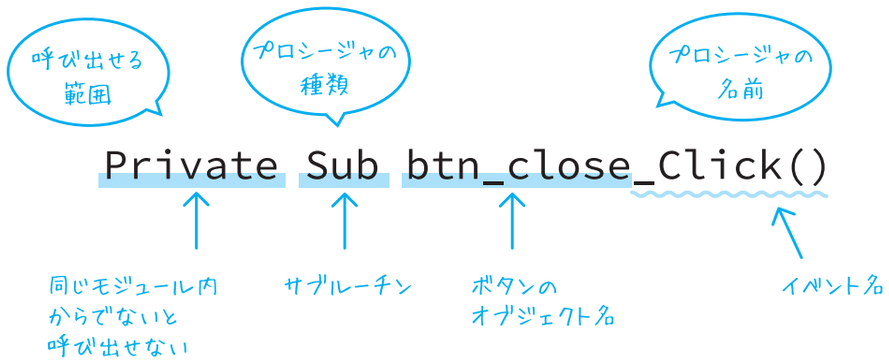
図26 「コードの表示」状態になった



ここに自動でプロシージャの枠が挿入されていますね。先頭行を詳しく見てみましょう（図27）。

フォームなどのオブジェクトを持つモジュールに「オブジェクト名_イベント名」という名称で作成されたプロシージャは、イベントプロシージャと認識されるという特徴があります。この名称であれば、このプロシージャは「btn_close」ボタンがクリックされたときに自動で起動します。

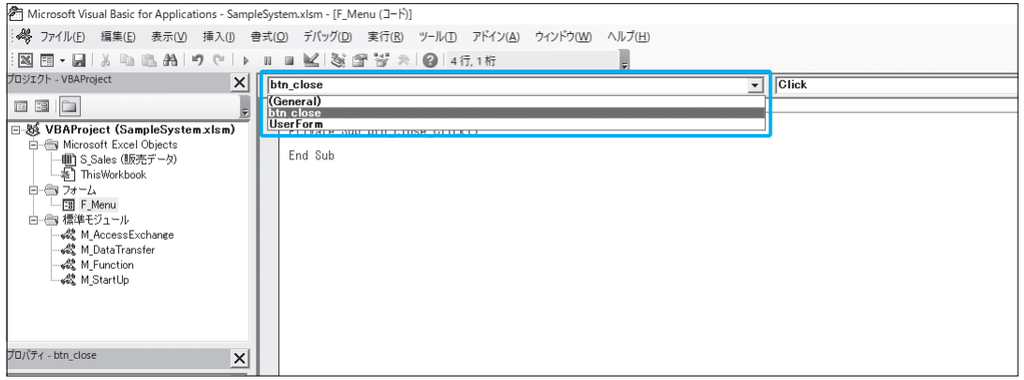
図27 自動挿入されたイベントプロシージャ



もちろん、イベントプロシージャと認識されるには、実際に存在するオブジェクトとイベントでないといけません。存在しないコントロールなど、記述に不備がある場合はジェネラルプロシージャと認識されます。

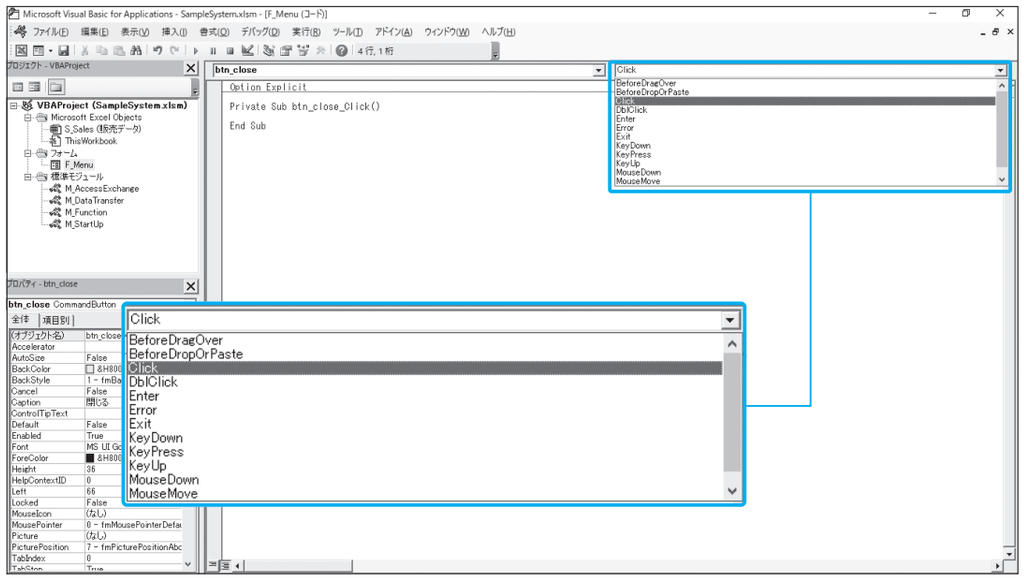
ちなみに、イベントはそのオブジェクトに対して複数存在します。コードウィンドウ上部、左側のドロップダウンリストには、現在選択されているモジュールに存在するオブジェクトの一覧が表示されます（図28）。

図28 存在するオブジェクト一覧



コマンドボタンである「btn_close」を選択すると、右側のドロップダウンリストには、そのオブジェクトに対応したイベントの一覧が表示されます（図29）。

図29 対応するイベント一覧



CHAPTER 8

8-5 「編集」フォームを拡張する

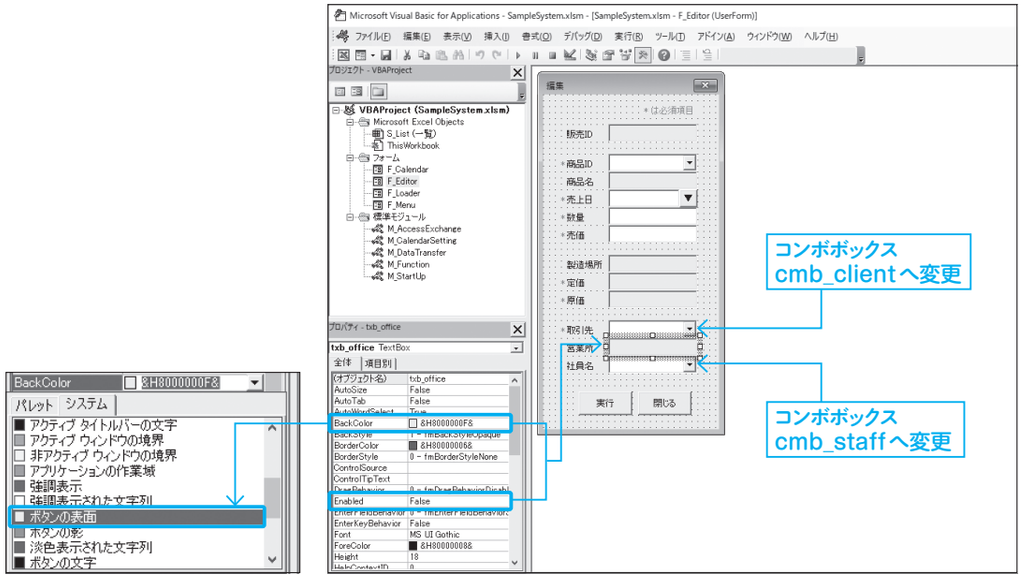
続いて、「編集」フォームを５つのテーブルに対応させていきます。

8-5-1 コントロールの変更

「F_Editor（編集）」フォームの「取引先」「社員名」テキストボックスをコンボボックスへ変更し、オブジェクト名を「cmb_client」「cmb_staff」とします。

また、「営業所」テキストボックスの「Enabled」を「False（編集不可）」に、「BackColor（背景色）」をグレー（「ボタンの表面」色）にしておきます（**図41**）。

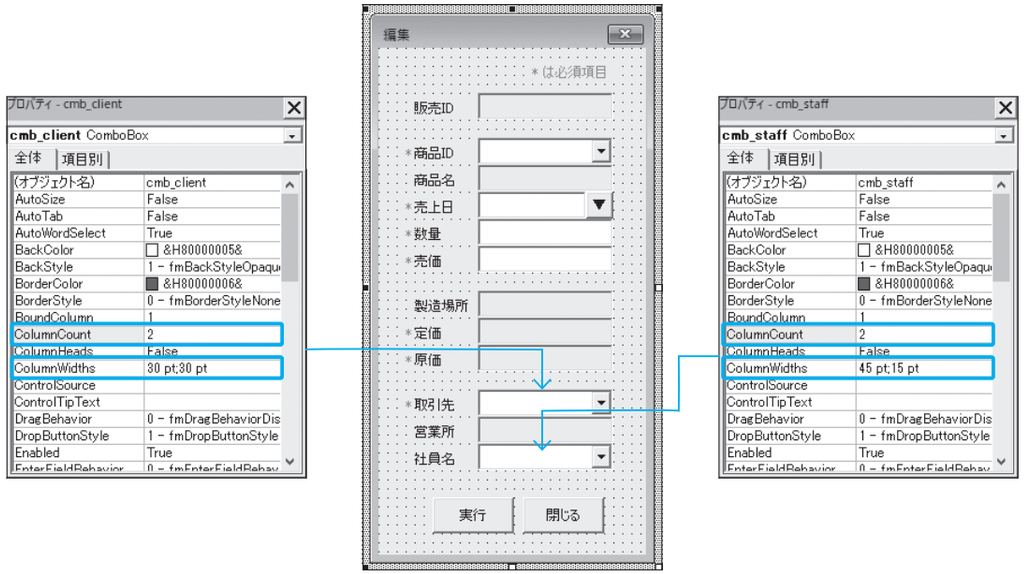
図41 フォームの改良



置き換えた「cmb_client」「cmb_staff」コンボボックスですが、テーブルの書込に必要な情報はID

でも、名称が出ていた方がユーザーは選びやすいので、2列表示させてみましょう。そのため**図42**のように、「ColumnCount（列数）」を「2」に、「ColumnWidths（列幅）」プロパティを「30pt;30pt」と「45pt;15pt」にそれぞれ設定します。

図42 コンボボックスの列数と列幅の設定



それではコードを修正していきましょう。

まずは「F_Menu（メニュー）」フォームモジュールからの動きを修正します。「メニュー」フォームの「更新」ボタンをクリックしたときの、現在選択されているセル位置のデータ転記部分です。「営業所」部分は「社員名」が選択されたときに自動入力される機能を付けるので、ここでは不要になります。テキストボックスからコンボボックスへの変更も反映します（**コード9**）。

コード9 「btn_update_Click」プロシージャ（「F_Menu」フォーム）

```
01 Private Sub btn_update_Click()
02   '##「更新」ボタンクリック時
       略
03
04   '「編集」フォームの代入と表示
05   With F_Editor
06     '変数設定
       略
07
```