Magic eBusiness Platform V9Plus

チュートリアル

Magic Plus
eBusiness
Platform™ V9

本書に記載の内容は、将来予告なしに変更することがあります。これらの情報について MSE（Magic Software Enterprises Ltd.）および MSJ（Magic Software Japan K.K.）は、いかなる責任も負いません。

本書の内容につきましては、万全を期して作成していますが、万一誤りや不正確な記述があったとしても、MSE および MSJ はいかなる責任、債務も負いません。

MSE および MSJ は、この製品の商業価値や特定の用途に対する適合性の保証を含め、この製品に関する明示的、あるいは黙示的な保証は一切していません。

本書に記載のソフトウェアは、製品の使用許諾契約書に記載の条件に同意をされたライセンス所有者に対してのみ供給されるものです。同ライセンスの許可する条件のもとでのみ、使用または複製することが許されます。当該ライセンスが特に許可している場合を除いては、いかなる媒体へも複製することはできません。

ライセンス所有者自身の個人使用目的で行う場合を除き、MSE または MSJ の書面による事前の許可なしでは、いかなる条件下でも、本書のいかなる部分も、電子的、機械的、撮影、録音、その他のいかなる手段によっても、コピー、検索システムへの記憶、電送を行うことはできません。

サードパーティ各社商標の引用は、MSE および MSJ の製品に対する互換性に関しての情報提供のみを目的としてなされるものです。

本書において、説明のためにサンプルとして引用されている会社名、製品名、住所、人物は、特に断り書きのないかぎり、すべて架空のものであり、実在のものについて言及するものではありません。

MSE および MSJ は、本製品の使用またはその使用によってもたらされる結果に関する保証や告知は一切していません。この製品のもたらす結果およびパフォーマンスに関する危険性は、すべてユーザが責任を負うものとします。

この製品を使用した結果、または使用不可能な結果生じた間接的、偶発的、副次的な損害（営利損失、業務中断、業務情報の損失などの損害も含む）に関し、事前に損害の可能性が勧告されていた場合であっても、MSE および MSJ、その管理者、役員、従業員、代理人は、いかなる場合にも一切責任を負いません。

初版　２００５年9月２６日

マジックソフトウェア・ジャパン株式会社

目　次

1. はじめに

Magic eBusiness Platform V9Plus をお使いいただきまして、ありがとうございます。Magic eBusiness Platform V9Plus は、高い生産性と保守性を誇る Magic パラダイムの歴史の上に、最新のテクノロジを融合して、マルチプラットフォーム、マルチ DBMS、マルチパラダイムの高度な RADD (Rapid Application Development and Deployment) ツールです。

Magic 製品により実現されるシステム形態は、スタンドアロンの小規模なものから大規模なクライアントサーバのオンラインシステム、およびサーバベースの Web アプリケーション、XML や Web サービス、J2EE 対応アプリケーション(サーバ、クライアント共)など、非常に広範囲にわたります。

本チュートリアルでは、Magic を始めて利用される方を対象に、スタンドアロンのオンラインアプリケーションをステップバイステップで作りながら Magic の基本を学んでいきます。

紙面の関係で、本書で紹介できるものは Magic の持つ広範囲な機能のごく一部でしかありません。より進んだ機能については、リファレンスマニュアル、開発者ガイド、添付サンプルアプリケーション、弊社ホームページ (http://www.magicsoftware.co.jp) の技術情報などを参照されることをお勧めいたします。

リファレンスマニュアル、開発者ガイドは、ファイルサイズの関係で体験版には添付されていませんが、弊社ホームページより PDF ファイルとしてダウンロードすることができます。

(http://www.magicsoftware.co.jp/mginfo/manual/#V9PLUS)

2. Magic の開発環境

インストールが終わったら、さっそく Magic を操作してみましょう。

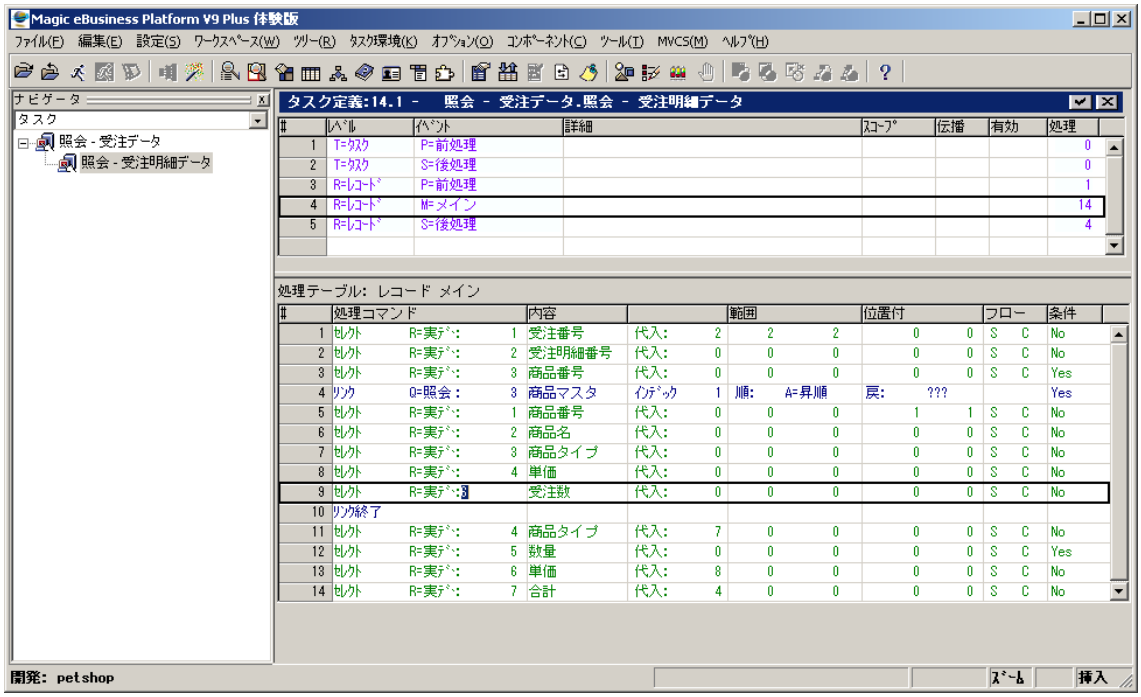


図 2-1 Magic の開発画面

2.1. Magic の起動

Magic を起動するには、スタートメニューから開始します。あるいは、デスクトップに作成されたショートカットをダブルクリックします。

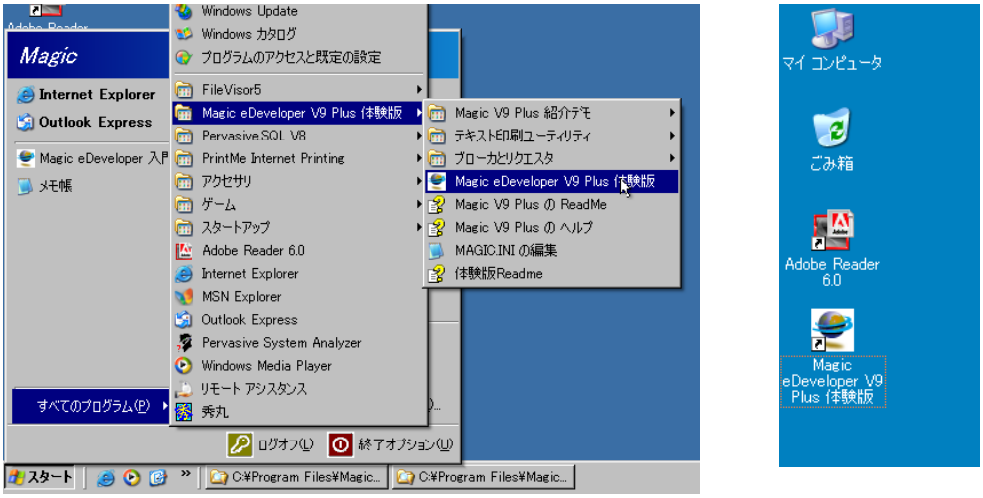


図 2-2 Magic の起動

Magic を起動した直後は、図 2-3 のような画面となっています。

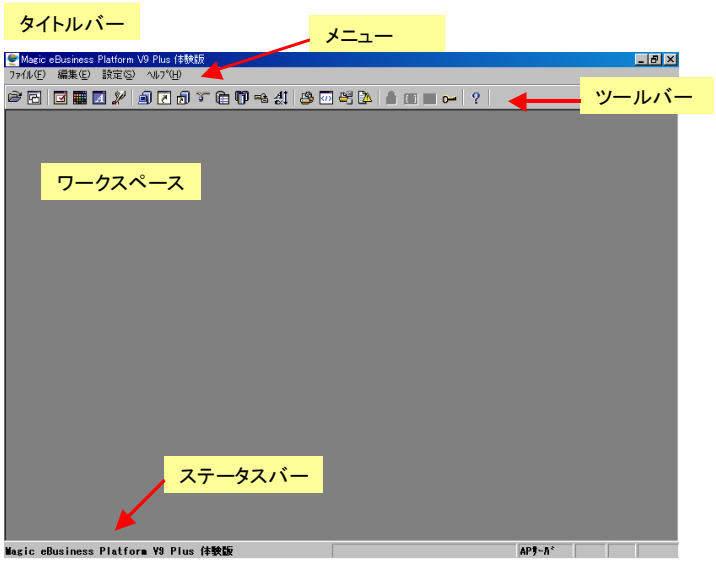


図 2-3 Magic 起動直後の初期画面

画面構成は、通常の Windows プログラムと同様、上からタイトルバー、メニュー、ツールバーがあり、ワークスペースがあって、最下段がステータスバーとなっています。

2.2. アプリケーションのオープン

Magic で作成するシステムのひとまとまりの単位は、**アプリケーション**と呼びます。アプリケーションはモデル定義、テーブル定義、プログラム定義、ヘルプやメニュー、権利設定の定義などを一まとめにしたもので、言語系で言う「プロジェクト」に相当するものです。

初期状態では、Magic のアプリケーションは開いていません。アプリケーションを開くには、メニュー「ファイル(F)」から「アプリケーション オープン(A)」を選びます。あるいは、ツールバーから、アプリケーション オープン アイコンをクリックします。

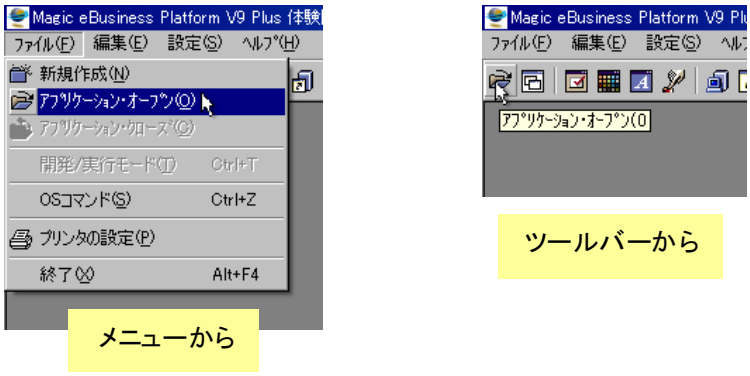


図 2-4 アプリケーションのオープン

登録されているアプリケーション一覧が開きますので、その中から開きたいアプリケーションを選択できます。体験版では、予めいくつかのサンプルアプリケーションが登録されており、その中から「ペットショップ」を選択してください。

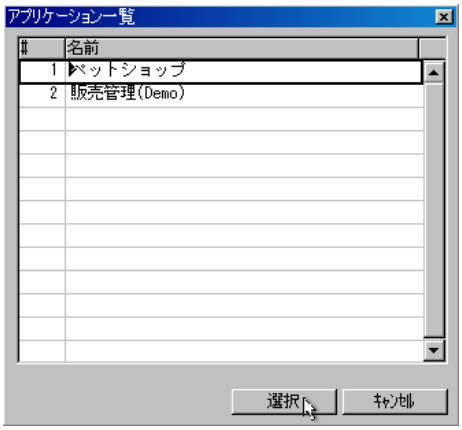


図 2-5 アプリケーション一覧

アプリケーションをオープンした直後は、開発画面が開きます[1]。

[1] これは体験版および Magic eDeveloper 製品(アプリケーション開発用製品)の場合です。Magic にはこのほかに、実行専用の Magic Client 製品があり、この製品ではオープン直後にすぐ実行画面となり、開発画面に切り替えることはできません。

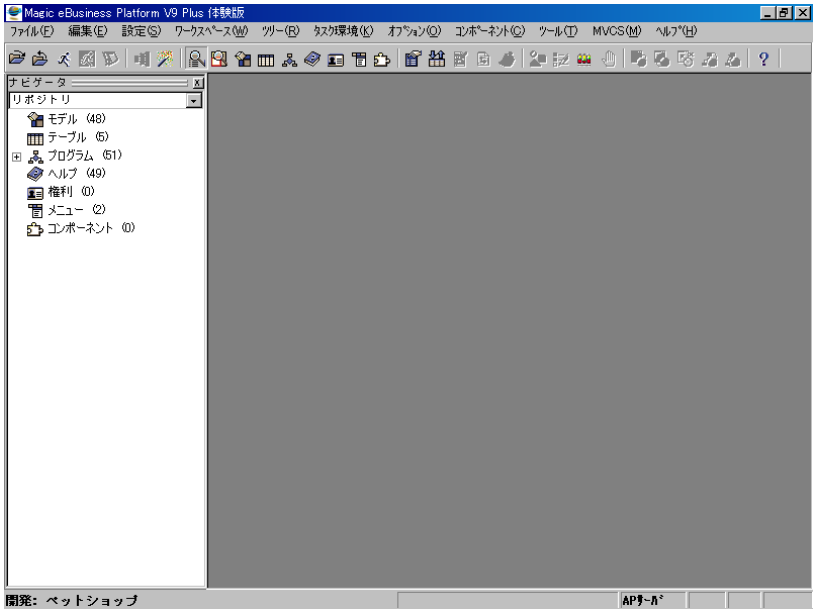

Magic eBusiness Platform V9 Plus 体験版
ファイル(F) 編集(E) 設定(S) ワークスペース(W) ツリー(R) タスク環境(K) オプション(O) コンポーネント(C) ツール(T) MVCS(M) ヘルプ(H)
ナビゲータ
リポジトリ
モデル (48)
テーブル (5)
プログラム (51)
ヘルプ (49)
権利 (0)
メニュー (2)
コンポーネント (0)
開発: ペットショップ

図 2-6 開発画面

2.3. 開発モードと実行モード

オープンしたアプリケーションには、**開発モード**と**実行モード**というふたつのモードがあります。
開発モードは、開発者がアプリケーションを開発するためのモードで、モデル、テーブル、プログラムなどの定義（**リポジトリ**と呼びます)を編集することができます。
実行モードというのは、開発モードで作成したプログラムを実際に実行するモードです。
両者の切り替えは、次のいずれかで行います。

- メニュー「ファイル(F)」から「T:開発/実行モード」を選ぶ
- 開発/実行モード(T) アイコンをクリックする。
- Ctrl + T キーを押す

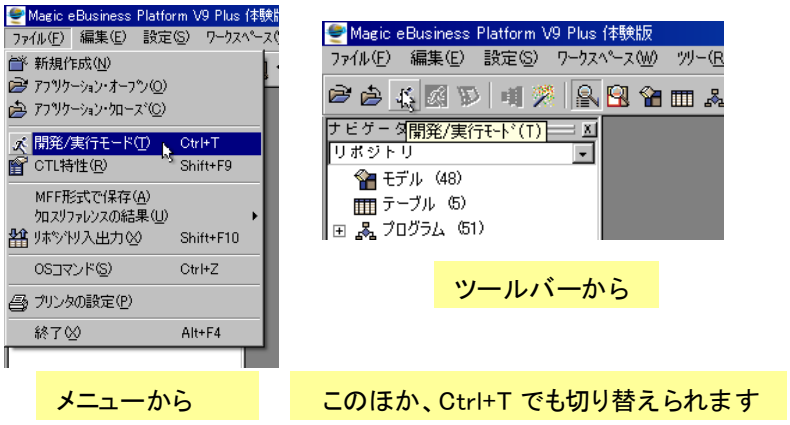


図 2-7 開発/実行モード切り替え

実行モードでは、アプリケーションで定義されたプルダウンメニュー（メニューバー)や、ポップアップメニュー（右マウスクリックで開く)が使えます。

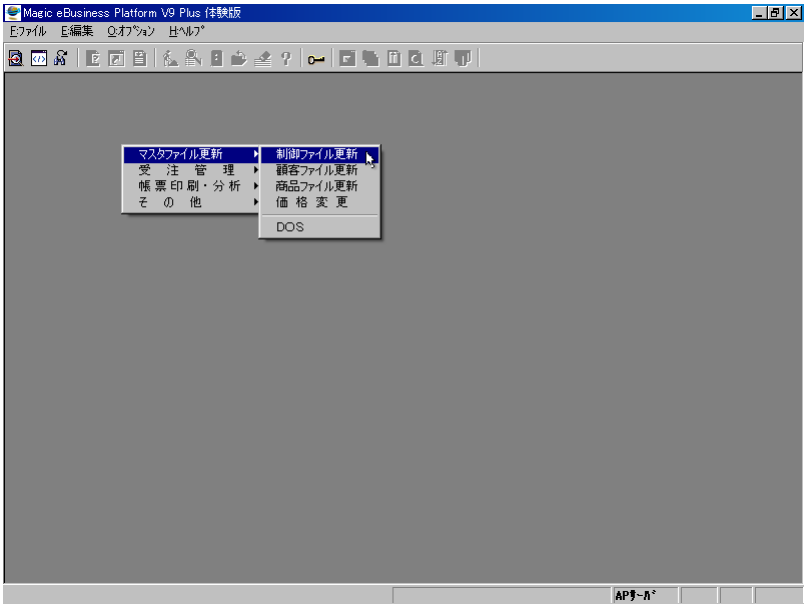


図 2-8 実行画面

実行画面から開発画面に戻るときにも、メニュー、アイコン、あるいは Ctrl + T キーで行います。

2.4. ナビゲータ

開発画面では、初期状態で、ワークスペースの左側に**ナビゲータ**が表示されます。ナビゲータというのは、アプリケーションで定義されているリポジトリをツリー形式で表示しているものです。

ナビゲータは最初左辺にドッキングしていますが、ナビゲータのタイトル部分をマウスでドラッグすると、フロート状態にしたり、右辺にドッキングさせたりすることもできます。画面の解像度などに合わせ、好きな形で利用してください。

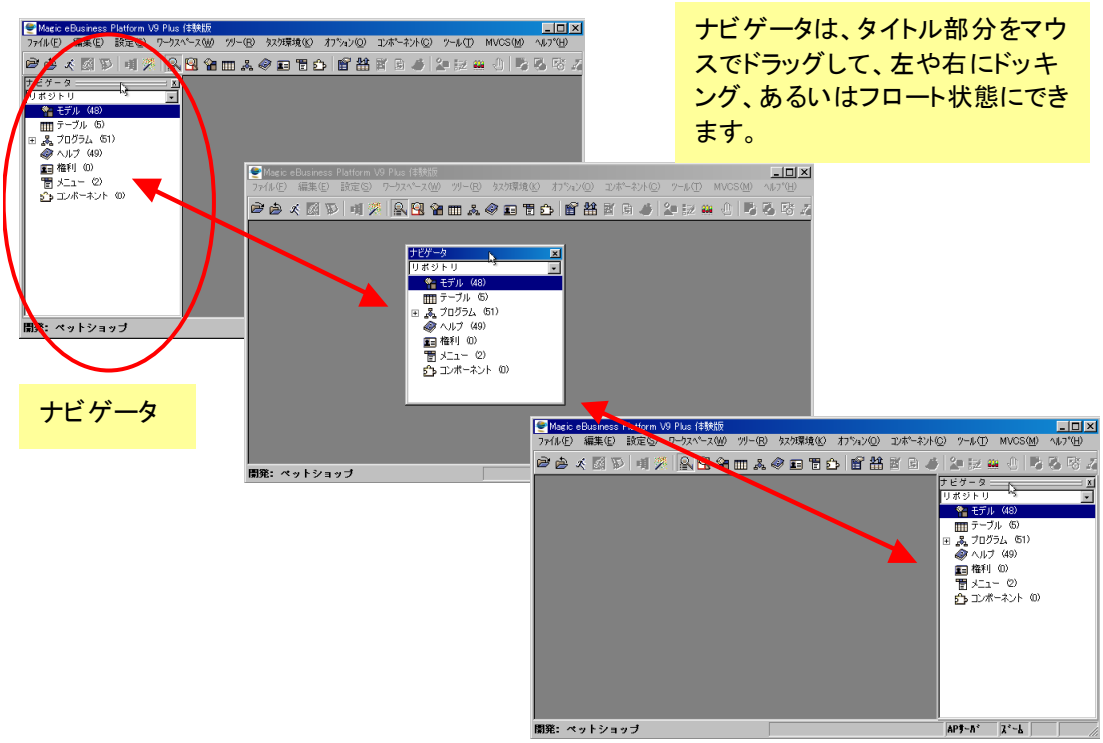


図 2-9 ナビゲータのドッキング

2.5. 特性シート

ここで、ナビゲータから「モデル」をクリックしてください。モデルリポジトリが開きます。ここでは、ナビゲータとモデルリポジトリの間に現れた「項目特性」というウィンドウを見てください。このウィンドウは、アプリケーションのオブジェクト(この場合モデル)の**特性**を表示するもので、**特性シート**と呼びます。

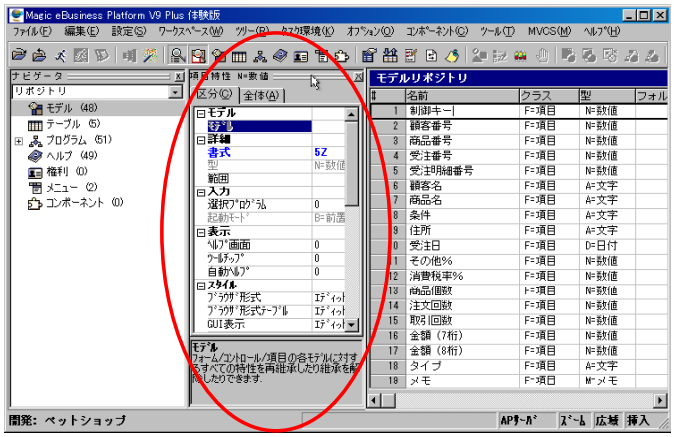


図 2-10 特性シート

モデルリポジトリおよび特性シートについては、後の章で追って説明しますが、大きな解像度のディスプレイを使っているのでない限り、ナビゲータと特性シートとが並んで表示されているのは、画面が狭くなって不便です。

このため、ナビゲータと特性シートとをドッキングさせることができます。特性シートのタイトル部分(「項目特性 N=数値」と書いてあるあたり)を、マウスでドラッグし、ナビゲータ上にまで持ってきて、マウスを放します。すると、ナビゲータと特性シートとがドッキングし、ひとつのウィンドウで表示されるようになります。表示の切り替えを行いたいときには、タブで行います。

タイトル部分をドラッグしてドッキングできます。

タブで表示を切り替えられます。

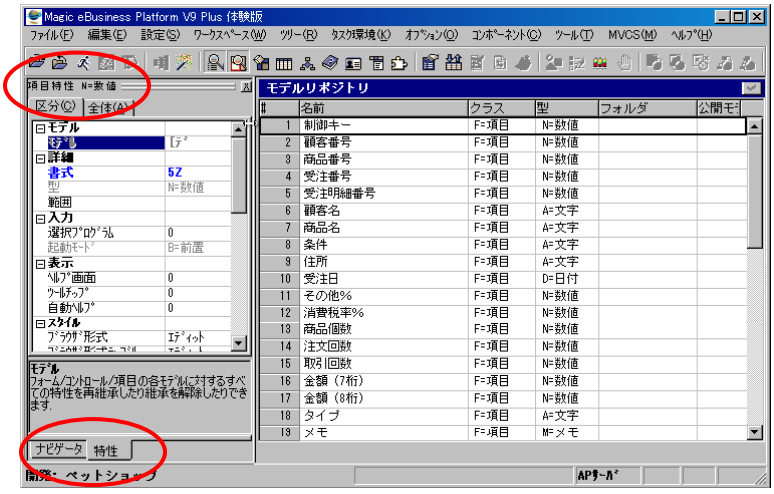


図 2-11 ナビゲータと特性シートとのドッキング

参考： これとは逆に、ドッキングしているナビゲータと特性シートとを切り離したい場合には、Ctrl キーを押しながら、タイトル部分をドラッグします。

2.6. 各種リポジトリ

Magic のアプリケーション定義は、各種のリポジトリに格納されます。Magic のアプリケーション定義は、次のような各種のリポジトリが格納されます。

- モデル
- テーブル
- プログラム
- ヘルプ
- 権利
- メニュー
- コンポーネント

リポジトリを開くにはいくつかの方法がありますが、例えば、モデルリポジトリを開く場合には、次のいずれかの方法によります。

- ナビゲータから： ナビゲータ上の「モデル」をクリックします。
- メニューから： 「ワークスペース(W)」メニューから、「モデル(O)」を選びます。
- ツールバーから： ツールバーの「モデル(O)」アイコンをクリックします。
- キーボードで： Shift+F1 キーを押します。

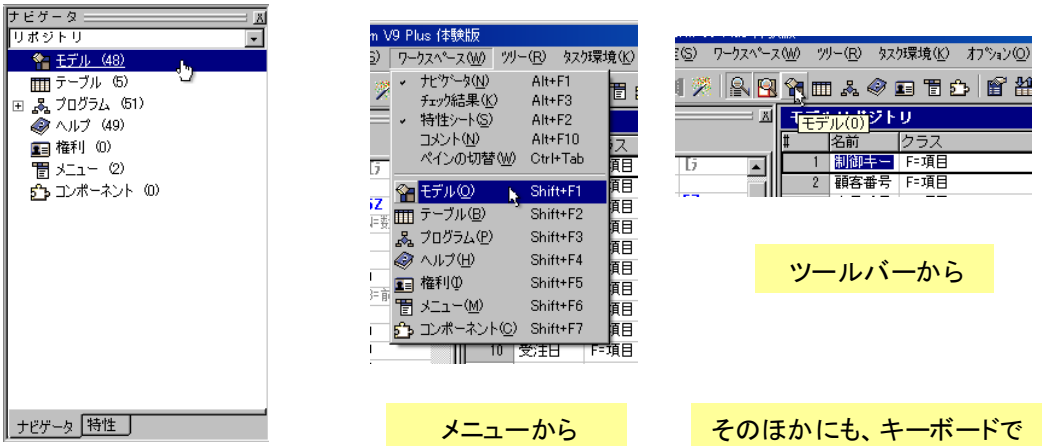


図 2-12 モデルリポジトリの開き方

他のリポジトリの開き方も同様に、ナビゲータ、メニュー、ツールバーおよびキー入力で開くことができますので、詳しい説明は省略します。

2.7. 基本キー操作

Magic 内でのキー操作は、Windows の標準的なキー操作に準じていますが、ここで、Magic に特有で、知っておくと便利なキー操作をいくつか説明しておきましょう。これは Magic の中でほぼどこにでも共通なキー操作です。

動作	キー	説明
クローズ	ESC	ウィンドウを閉じるときや、実行中のプログラムを止める時に使います。
OK	Enter	確認してウィンドウを閉じる場合や、フォーカスのあるボタンを選択する場合など、「OK」の意味で Enter を使います。
ヘルプ	F1	コンテキストに応じたヘルプメニューが表示されます。
取消	F2	現在入力した内容が、確定される前ならば、取り消して元の値に戻します。
行削除	F3	テーブル形式で表示されているデータの場合、現在カーソルのある行を削除します。
行追加	F4	現在カーソルのある行の次に、新しい空白行を追加します。
ズーム	F5、 ダブルクリック	ある項目で設定可能な一覧を表示して選択したり、より詳細な設定を行う場合に別のダイアログを開いたりします。
広域表示	F6	表示領域が小さいが、実際にはより長いデータを格納できる場合、F6 で別ウィンドウが開き、データを入力できるようになります。

表 2-1 よく使うキー操作とその意味

このうち、**ズーム** (F5 またはマウスのダブルクリック)は、Magic で非常によく使います。すべての項目で可能であるわけではありませんが、ズーム可能な項目では、ステータスバーに「ズーム」と表示されるので、わかります。

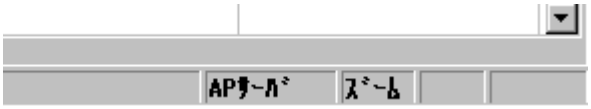


図 2-13 ズーム可能な項目でのステータスバー表示

2.8. コピー＆貼り付けのキー割り当て変更

現在の Windows の標準では、コピーは Ctrl + C 、貼り付けは Ctrl + V となっていますが、Magic のデフォルトの設定では、コピーは Ctrl + Insert 、貼り付けは Shift + Insert となっていますので、注意してください。

このキー割り付けが不便と感じられる場合には、次のようにして、キー割り付けを変更することができます。

1. メニュー「設定(S)」から「キーボード割付(K)」を選びます。キーボード割付ダイアログが開きます。

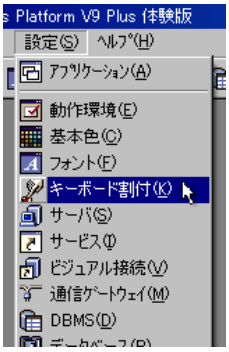


図 2-14 キーボード割付メニュー

2. 「アクション」欄が「貼り付け(A)」であるものを選びます(初期状態では、第 3 行目にあります)。
3. 「キー」欄でズーム(F5 あるいはダブルクリック)します。キー設定ダイアログが開きます。
4. Ctrl + V キーを押します。「Ctrl+V」と表示されます。
5. OK ボタンを押します。

図 2-15 「貼り付け」を Ctrl+V にマップ

6. 次に、「アクション」欄が「コピー(O)」である行に移ってください。初期状態では、第 91 行目にあります。
7. 同様に、「キー」欄からズームし、Ctrl+C を設定してください。

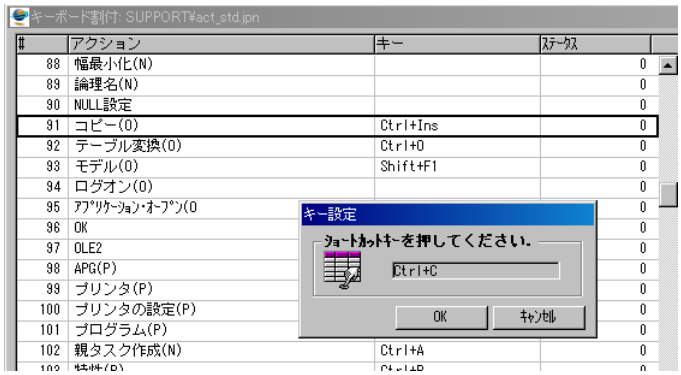


図 2-16 「コピー」を Ctrl+C にマップ

8. OK ボタンを押します。「ファイル名をつけて保存」ダイアログが出ます。
9. 「即時有効」欄は Yes にして、OK ボタンを押してください。

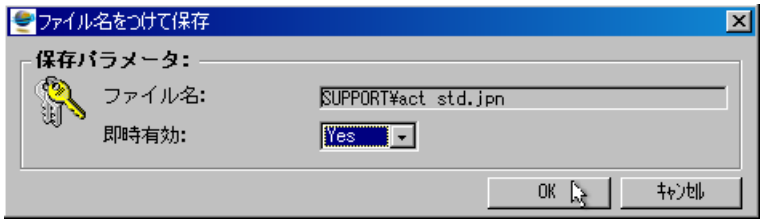


図 2-17 保存ダイアログ。即時有効は Yes にして保存。

これで、ショートカットキーの割り当てが変わります。

参考：

- コピーと貼り付けに限らず、他のキー割り付けも、同様の操作により変更することができます。
- このようにキーの割り付け変更を行うと、通常 Ctrl+C、Ctrl + V に割り当てられているアクションが利用できなくなりますので、注意してください。詳細は省略しますが、例えば、Ctrl+C はプログラムリポジトリでの「タスク制御」ダイアログを開くためと、実行時に登録モードに変えるため、また、Ctrl+V はプログラムリポジトリで変数テーブルを開くために使えますが、これらのキーが効かなくなります。

3. 新規アプリケーションの登録

本章以降では、新規に空のアプリケーションを作成し、そこから簡略化したペットショップデモを作成していくことを通し、Magic のアプリケーションの作成方法を学んでいきます。

本章では、まず、新規にアプリケーションを作成するところからはじめます。

3.1. 新規にアプリケーションを登録する

新規に空のアプリケーションを作成するには、以下の手順によります。

1. メニュー 「ファイル(F)」→「新規作成(N)」を選びます。
2. 「新規アプリケーション」ダイアログで、「アプリケーションの名前」を「petshop」とします。
3. 「開始時に初期ヘルプを表示」のチェックをはずします。

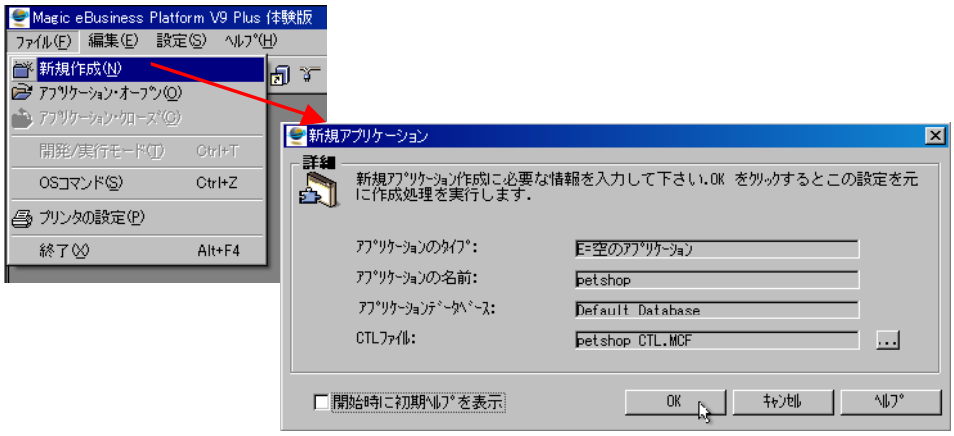


図 3-1 新規アプリケーションダイアログ

4. 「OK」ボタンを押します。空のアプリケーションが新規に作成され、開発モードになります。

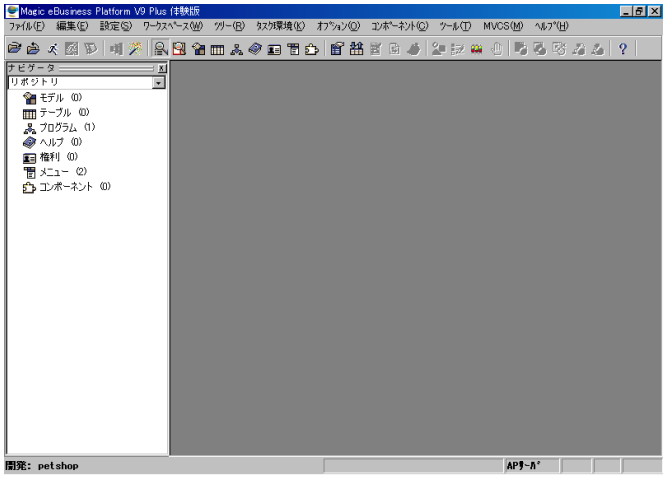


図 3-2 開発モード画面

3.2. アプリケーションファイルの位置を確認する

Magic のアプリケーションは、MCF (Magic Control File)というファイルにすべて格納されます。このファイルがどこにあるのか確認しましょう。

1. まず、ツールバーから「アプリケーションクローズ」アイコンをクリックして、アプリケーションをクローズします。

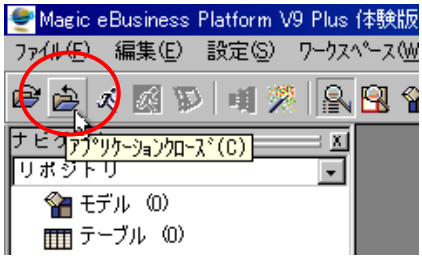


図 3-3 アプリケーションのクローズ

2. ツールバーから、「アプリケーション」アイコンをクリックすると、アプリケーションテーブルが開きます。

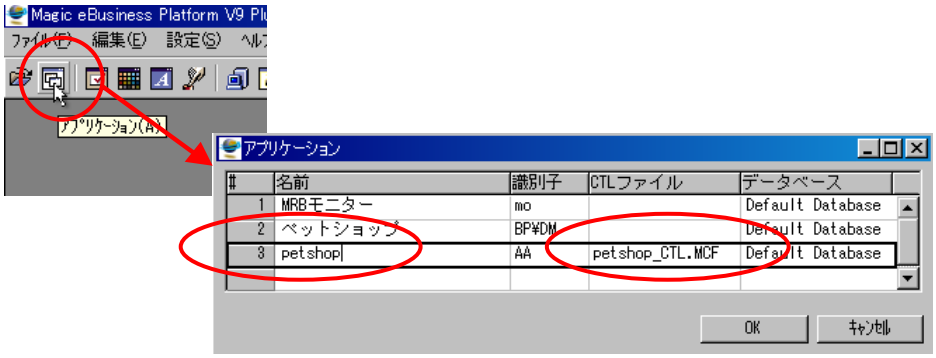


図 3-4 petshop アプリケーションを開く

3. アプリケーション「petshop」の「CTL ファイル」欄を見ると、「petshop_CTL.MCF」となっています。ここには MCF ファイル名を指定しますが、ファイル名のみを指定した場合には、Magic ディレクトリに作成されます。OS のフルパス名を指定することもできます。
4. エクスプローラを開き、Magic ディレクトリ C:¥Program Files¥Magic¥eDeveloperPlusTrial を開きます[2]。ここに、PETSHOP_CTL.MCF が作られていることを確認してください。

[2] これは、デフォルトでインストールした場合のディレクトリ名です。もしインストール時にディレクトリを変更した場合には、そこを開いてください。

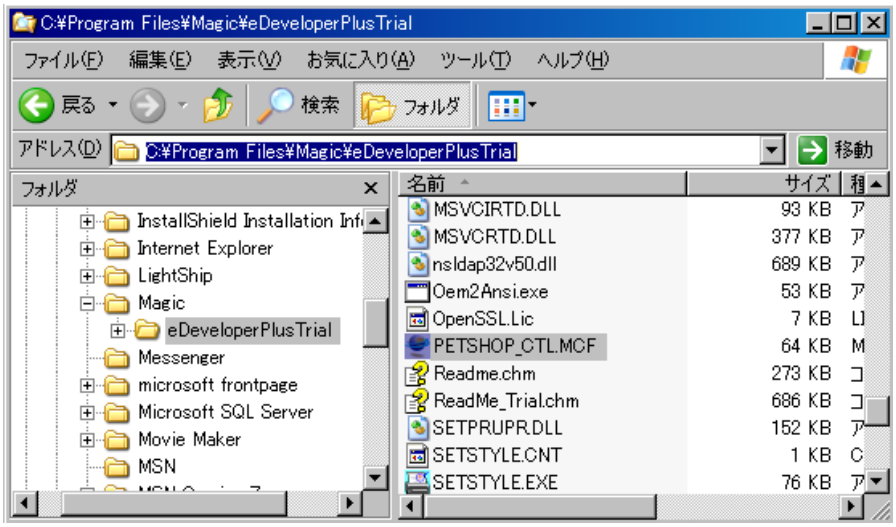


図 3-5 petshop アプリケーションの MCF ファイル

参考： このファイルは、Pervasive SQL のファイル形式で作成されているバイナリファイルですので、テキストエディッタなどで開いてもアプリケーションの内容を見ることはできません。

4. データ定義

アプリケーションの基本となるのは、データ定義です。本章では、Magic でのデータ定義方法を説明します。

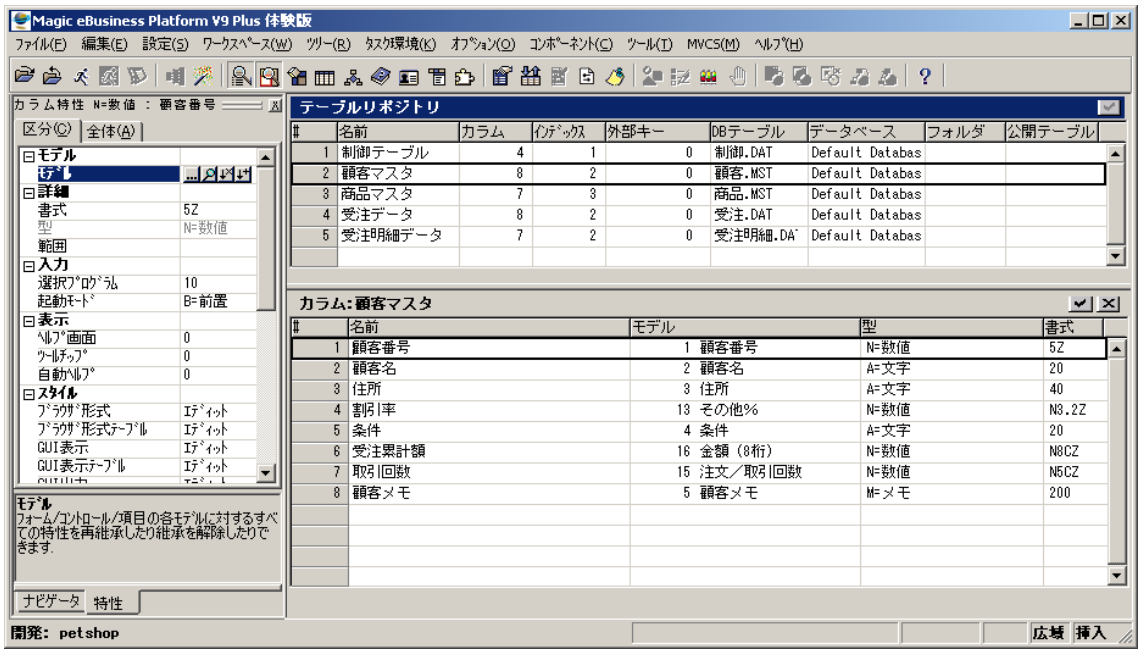


図 4-1 Magic のテーブルリポジトリ

参考情報： 本章の内容については、リファレンスマニュアルにより詳細な情報がありますので参考にしてください。

モデルリポジトリ全般について	第 3 章 モデル
データ項目について	第 3 章 モデル → データ項目
書式について	第 3 章 モデル → 書式
項目モデルについて	第 3 章 モデル → 項目モデル特性
テーブルリポジトリについて	第 4 章 テーブル
APG について	第 19 章 ユーティリティ → APG (自動プログラムジェネレータ)

4.1. データ型

Magic が扱える基本データ型には、以下のようなものがあります。

- **文字型およびメモ型**： 任意の文字列を格納できます。文字型とメモ型の違いは、DBMS に格納する場合の形式の違いで、文字型は固定長で格納し、メモ型は可変長で格納します。いずれも最大長は 32,000 バイトです。
- **数値型**： 最大有効桁数が 18 桁までの数値を扱うことができます[3]。内部では独自形式の 10 進数で格納されているので、小数点以下の桁のある数値であっても、誤差は生じません。
- **論理型**： 論理値 「真」あるいは「偽」を格納します。
- **日付型**： 日付を格納します。西暦 1 年 1 月 1 日を基点としますが、特別な値として、0000/00/00 という日付も可能です。
- **時刻型**： 時間、分、秒のデータを格納します。最大 99 時間 59 分 59 秒まで格納できます。内部では数値型として保持しています。

このほかに、BLOB (Binary Large Object Block)、ベクトル型(配列)、ActiveX 型などの特殊型がありますが、本書では扱いません。

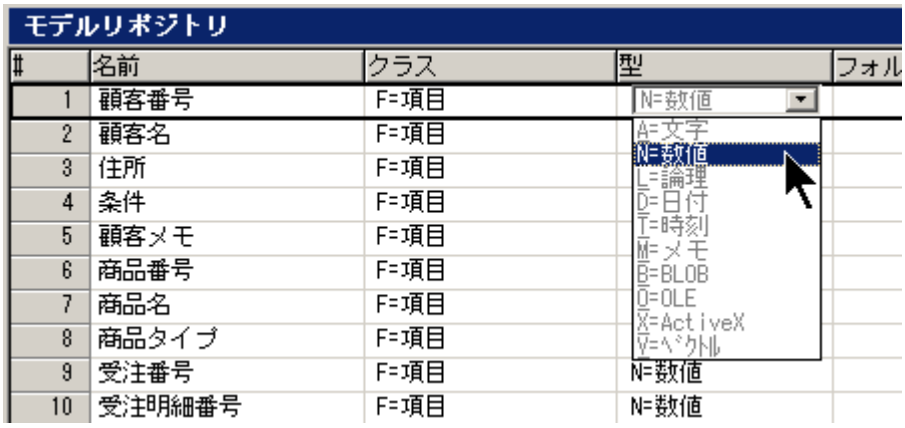


図 4-2 Magic がサポートするデータタイプ（モデルリポジトリでの定義）

[3] オプションにより、最大 38 桁までとすることもできますが、ここでは省略します。

4.2. 書式

各データ項目の定義と表示のために、「書式」が使われます。Magic が扱える代表的な書式には、次のようなものがあります。

表 4-1 Magic の書式例

データ型	書式例	意味
文字型およびメモ型	10	10 バイトの文字列（全角文字の場合には 5 文字まで)
	U5 UUUUU	半角大文字 5 つ U は半角大文字の意味。
	#5 #####	半角数字 5 桁 # は半角数字の意味。
数値型	10.3	小数点以上 10 桁、小数点以上 3 桁（合計 13 桁）の非負の数。
	N10.3	上と同じだが、負数も許す。 N は負数可を意味する。
	N8CZ	小数点以上 8 桁の非負の整数。 C は 3 桁ごとにカンマを入れることを意味する。 Z は値がゼロの場合には空白とすることを意味する。
論理型	5	True または False を表示
日付型	YYYY/MM/DD ####/##/##	西暦 4 桁、月 2 桁、日にち 2 桁の形式。
	JJJJYY 年 MM 月 DD 日	JJJJ は元号(「平成」など)、 MM, DD はそれぞれ月、日にちを意味する。 「年」「月」「日」などの文字は、そのまま表示されるが、DBMS には格納されない。
時刻型	HH:MM:SS	時間、分、秒をそれぞれ 2 桁つづで表示。

4.3. モデルリポジトリの定義

4.3.1. Magic のモデルとは？

Magic アプリケーションの作成にあたっては、データテーブル定義に先立って、**モデル**を定義しておくと便利です。

Magic のモデルというのは、アプリケーションで使用するデータのプロトタイプであり、プログラミング言語でのタイプと同じ概念です。たとえば、「顧客番号」「商品名」などをモデルとして定義することができます。モデルを使うことにより、

- 同様の定義をアプリケーションの中で何回も繰り返し定義する手間が省け、間違いの可能性が減ります。
- データ項目の意味がわかりやすくなり、アプリケーションの見通しが良くなります。
- データ定義に変更があった場合(例えば、顧客番号を 5 桁から 7 桁に拡張するなど)、モデルを一箇所変更するだけで、アプリケーション全体に有効となるので、保守性が高まります。

などの利点があります。

Magic の場合には、プログラミング言語のタイプのようにデータ型やサイズを定義するだけでなく、データ項目の書式、表示形式、デフォルト値、NULL の可否、可能な値の範囲や列挙、選択プログラムなど複雑なプロパティも定義することができます。

更には、Magic のモデルはデータ項目に止まらず、フォーム上のコントロール(エディットコントロール、コンボボックス、リストボックス、チェックボックス、ラジオボタン、プッシュボタン、テーブルコントロールなど)、フォームなど、アプリケーションで使う広範囲なオブジェクトに対してプロトタイプを定義する高度なメカニズムを提供します。

ここでは、アプリケーションで使うデータ項目に対するプロトタイプとしてのモデル(項目モデル)だけを説明します。

4.3.2. ペットショップデモで使うモデル

ペットショップデモで使うモデルには、以下のようなものがあります。

表 4-2 ペットショップデモで使う項目モデル

#	モデル名	データ型	書式
1	顧客番号	数値	5Z
2	顧客名	文字	20
3	住所	文字	40
4	条件	文字	20
5	顧客メモ	メモ	200
6	商品番号	数値	5Z
7	商品名	文字	20
8	商品タイプ	文字	AU
9	受注番号	数値	3Z
10	受注明細番号	数値	3Z
11	受注日	日付	YYYY/MM/DD

12	消費税率%	数値	3.2Z
13	その他%	数値	N3.2Z
14	商品個数	数値	N5CZ
15	注文/取引回数	数値	N5CZ
16	金額（8 桁）	数値	N8CZ
17	合計（10 桁）	数値	N10CZ
18	制御キー	数値	5Z
19	顧客へのメッセージ	メモ	200

ここではデータ型と書式のみ示していますが、後の章で、選択プログラムや表示形式などもモデルで設定するようにしていきます。

4.3.3. Magic でのモデルの定義方法

Magic では、モデルは、「モデルリポジトリ」に定義します。
アプリケーションを開いていなければ、ここで最初に開いてください。

アプリケーションの開き方

Magic アプリケーションは、次のいずれかの方法により開くことができます。

- 「ファイル」メニューより、「アプリケーションオープン(O)」を選ぶ。
- ツールバーから、「アプリケーションオープン(O)」を選ぶ。

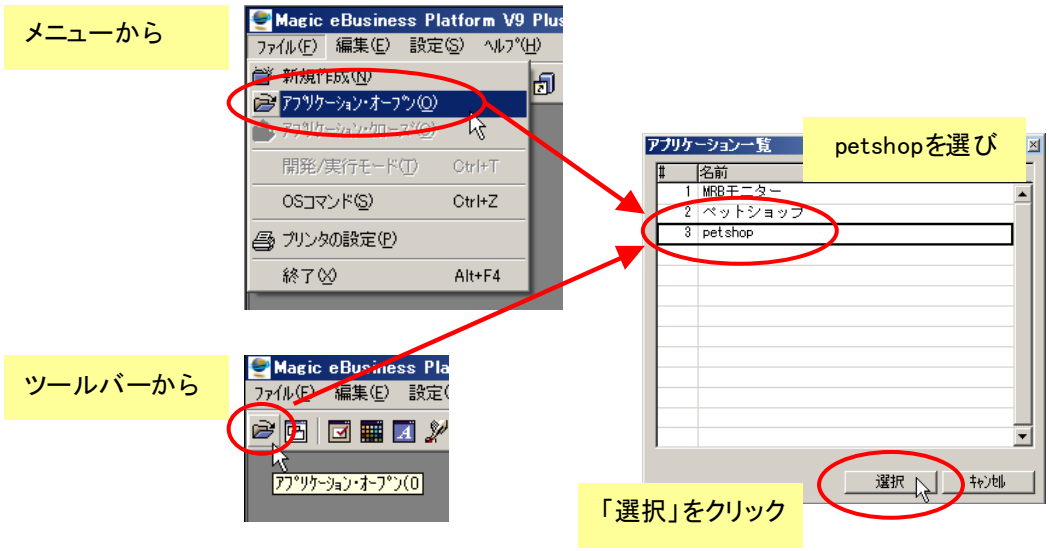


図 4-3 アプリケーションのオープン

モデルリポジトリの開き方

モデルリポジトリは、次のいずれかの方法により開けます。

- **メニューから：** 「ワークスペース」メニューから「モデル(O)」を選びます。
- **ツールバーから：** ツールバーで「モデル(O)」をクリックします。
- **ナビゲータから：** ナビゲータから「モデル」をクリックします。
- **キー操作により：** Shift+F1 を押します。

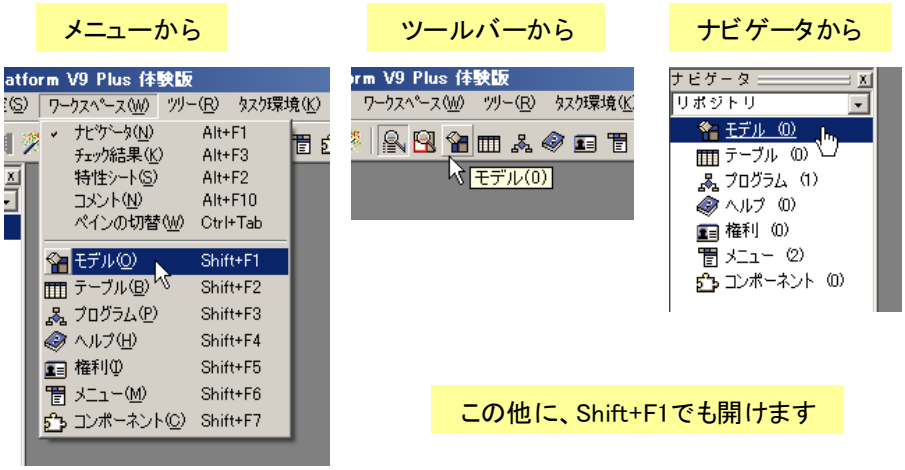


図 4-4 モデルリポジトリの開き方

モデルリポジトリでのモデルの定義のしかた

モデルは次の手順で作成します。ここでは、顧客番号（数値型、書式は「5Z」）を例にしています。

1. F4 で新規行を作成します。
2. 名前を記入します。
3. 「クラス」は「F=項目」に
4. 「型」は「N=数値」を指定します。
5. マウスでプロパティシートの「書式」に移ります。(Ctrl+P でも移動します)
6. 「書式」に「5Z」を指定します。

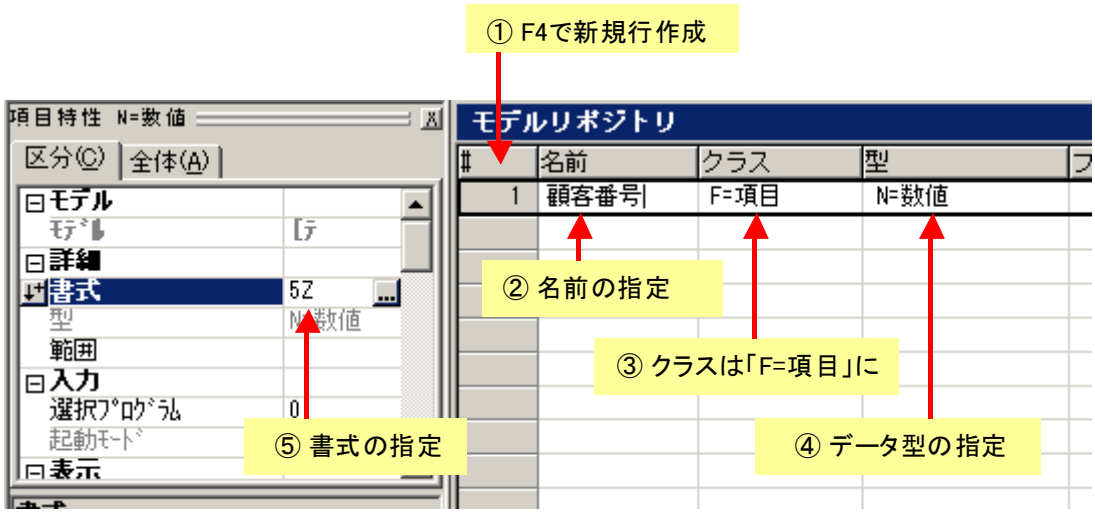


図 4-5 モデルの定義

同様な要領で、「表 4-2 ペットショップデモで使う項目モデル（29 ページ）」に挙げたすべてのモデルを定義してください。

4.4. アプリケーションで使うテーブル

ここではペットショップデモで使う、以下のテーブルについて、テーブル定義を説明します。

表 4-3 ペットショップデモで使うテーブル

#	テーブル名	説明
1.	制御テーブル	アプリケーション全体で共通なパラメータ類を格納します。
2.	顧客マスタ	顧客に関する情報を格納したマスターテーブル
3.	商品マスタ	商品に関する情報を格納したマスターテーブル
4.	受注データ	注文に関するデータを格納する。1 件の注文につき、1 レコードが対応する。
5.	受注明細データ	各注文について、注文の明細を格納する。1 件の注文の 1 商品について、1 レコードが対応する。

4.4.1. テーブルリポジトリの開き方

テーブルリポジトリは、次のいずれかの方法により開きます。

- **メニューから：** 「ワークスペース」メニューから「テーブル(B)」を選びます。
- **ツールバーから：** ツールバーで「テーブル(B)」をクリックします。
- **ナビゲータから：** ナビゲータから「テーブル」をクリックします。
- **キー操作により：** Shift + F2 を押します。

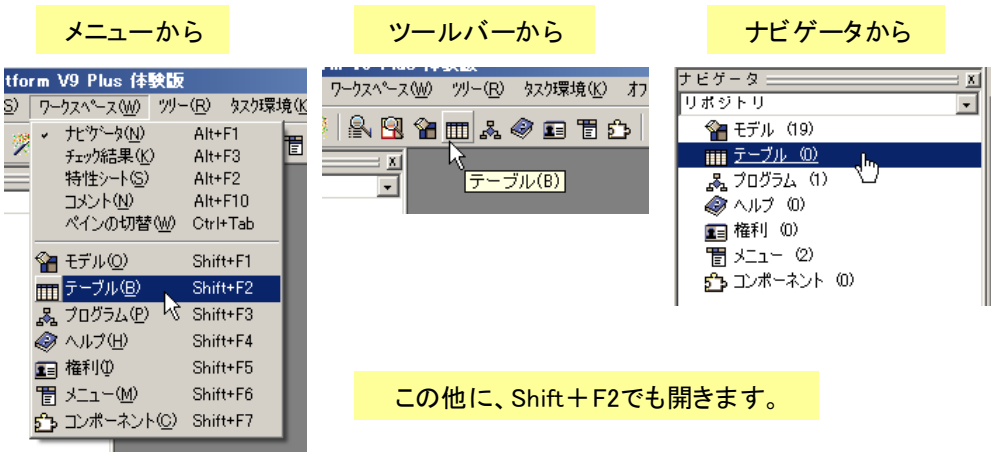


図 4-6 テーブルリポジトリの開き方

4.4.2. 制御テーブルの定義

次に、テーブルをひとつづつ定義していきます。最初に制御テーブルを定義します。

制御テーブルは、アプリケーション全体で共通なパラメータ類を格納します。このテーブルには、レコードがひとつだけあり、キーの値は 1 です。

テーブルリポジトリでのテーブルの定義は、以下のようにして行います。

(1) テーブルの基本定義

制御テーブルの基本定義は、以下の通りです。

表 4-4 制御テーブルの基本定義

名前：	制御テーブル
DB テーブル名：	制御.DAT

これは、テーブルリポジトリで以下のように設定します。

1. F4 で新規行を作成します。
2. 「カラム」「インデックス」「外部キー」欄はスキップします。(後で定義します)
3. 「名前」にテーブル名「制御テーブル」を記入します。
   「名前」は、Magic アプリケーションの中でだけ使うもので、空白や特殊文字などの文字も自由に使えます。
4. DB テーブルには、「制御.DAT」という名前を指定します。
   「DB テーブル名」は、OS 上に作成されるファイル名となります。ファイルのフルパスを指定することもできますが、パス指定がない場合には、現在の作業ディレクトリに作成されます。

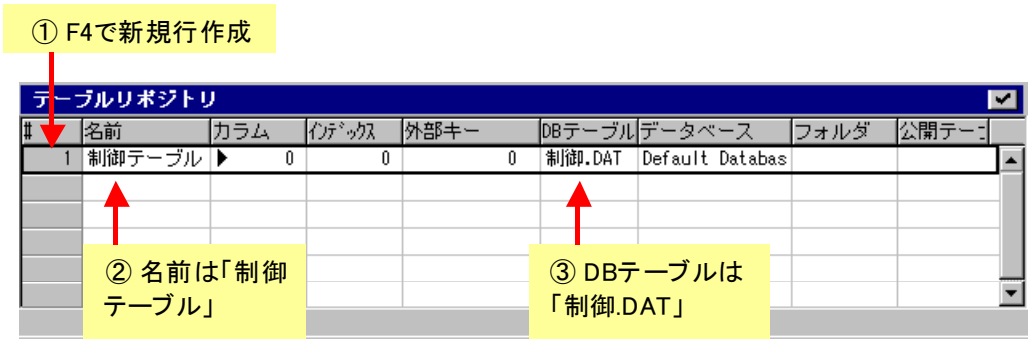


図 4-7「制御テーブル」の基本定義

カラムの定義

制御テーブルのカラム定義は、以下の通りです。

表 4-5 制御テーブルのカラム定義

制御テーブル カラム定義					
#	カラム名	モデル番号	モデル名	データ型	書式
1	制御キー	18	制御キー	数値	5Z
2	消費税率	12	消費税率%	数値	3.2Z
3	最終受注番号	9	受注番号	数値	3Z
4	顧客へのメッセージ	19	顧客へのメッセージ	メモ	200

テーブルの全カラムを、以下の手順で定義します。

1. 「カラム」欄にカーソルがある状態で、ズーム(F5 を押す)します。カーソルはカラムテーブルに移動します。
   マウスクリックで「カラム」テーブルをクリックして、カラムテーブルに移動することもできます。

2. F4 で新規行を作成します。「名前」欄はスキップします。
3. 「モデル」カラムでズームすると、一覧が出てきます。
4. 一覧から、「制御キー」モデルを選択します。
   カラムテーブルにモデルの番号(18)が設定され、名前、型、書式などには、モデルで定義されているのと同じ値が自動的に設定されます。
5. 以下同様にして、制御テーブルのすべてのカラムを定義します。

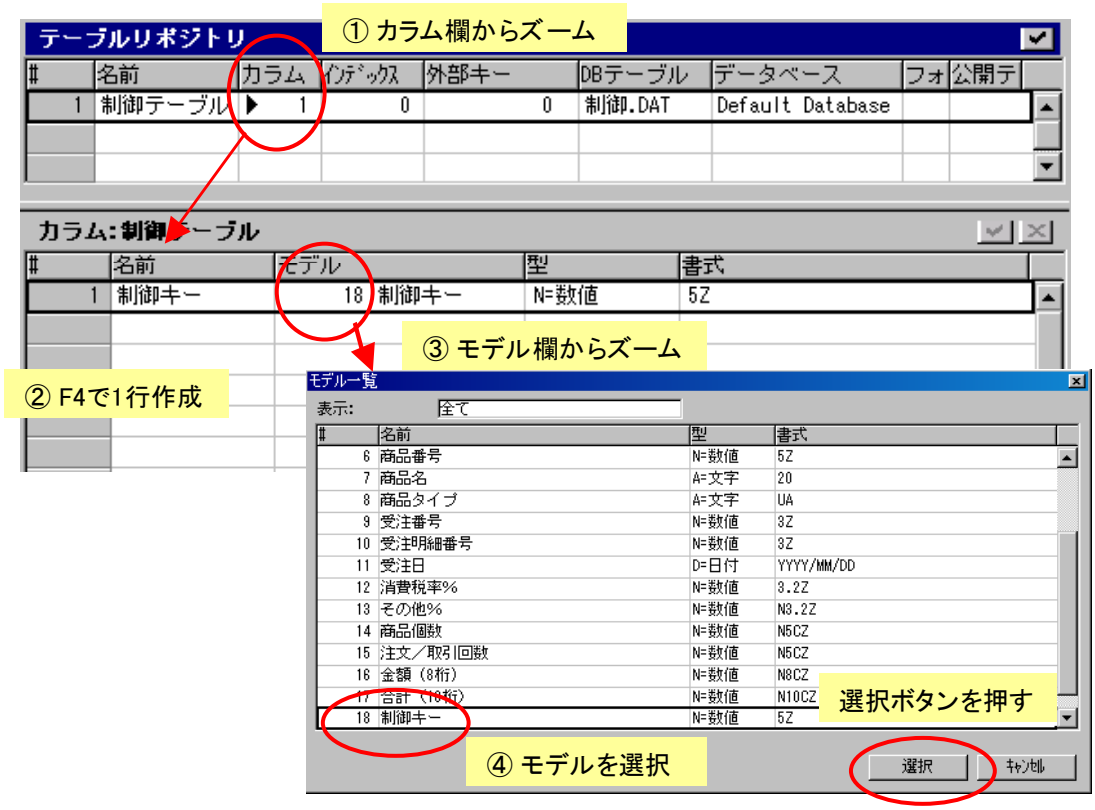


図 4-8 モデルを使ってカラムの定義

参考：

- 「最終受注番号」のように、モデルの名前とカラムの名前が異なる場合には、モデルを設定した後で、カラムの「名前」欄を変更してください。
- 「モデル」欄では、ズームして一覧から選択する変わりに、モデルの番号を直接入力することもできます。モデル番号を覚えている場合には、入力をすばやく行うことができます。

インデックスの定義

制御テーブルには、以下のようなインデックスをひとつだけ定義します。

制御テーブル インデックス定義			
#	インデックス名	タイプ	セグメント
1	制御キー	重複不可	1 制御キー

表 4-6 制御テーブルのインデックス定義

セグメントというのは、インデックスを構成するカラムです。制御テーブルでは制御キーカラムひとつだけからインデックスが定義されていますが、複数のカラムを組み合わせてインデックスを定義することも可能です。

テーブルのインデックスは、以下の要領で定義してください。

1. 「インデックス」欄に移り、ズームします。カーソルがインデックステーブルに移ります。
2. F4 で新規行を作成します。
3. 「名前」にインデックス名として「制御キー」と記入します。
   この名前は Magic の内部でだけ使うもので、空白特殊文字など含め自由に設定することができます。
4. 「タイプ」欄には、インデックスのタイプ (重複不可／可)を指定します。
   デフォルトでは重複不可です。
5. 「名前」欄からズームして、「セグメント」テーブルに入り、F4 で新規行を作成します。
6. キーに含めるカラムのカラム番号 1 を指定します。
   「カラム」欄でもう一度ズームすると右のカラム一覧に行くので、ここから選択しても良いし、また、カラム番号を直接入力しても構いません。

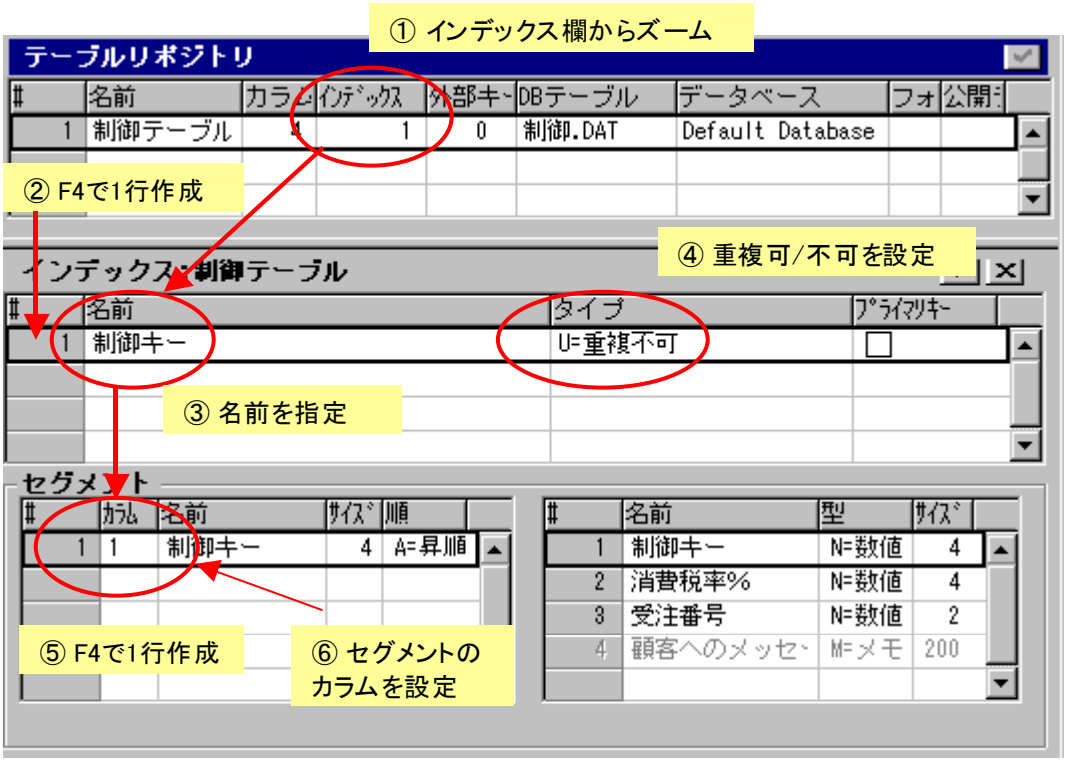


図 4-9 制御テーブルのインデックス定義

テーブルのチェック

テーブルの定義が終わったら、定義に文法上の誤りがないかを、チェック機能を使ってチェックしましょう。

1. テーブルリポジトリの「制御テーブル」の行にカーソルを置きます。
2. メニュー「オプション(O)」→ 「チェック(S)」により、チェック機能を呼び出します。
   メニューの代わりに、F8 キーを押してもできます。
3. ステータス行に「テーブルは正常です」と表示されれば、正常です。

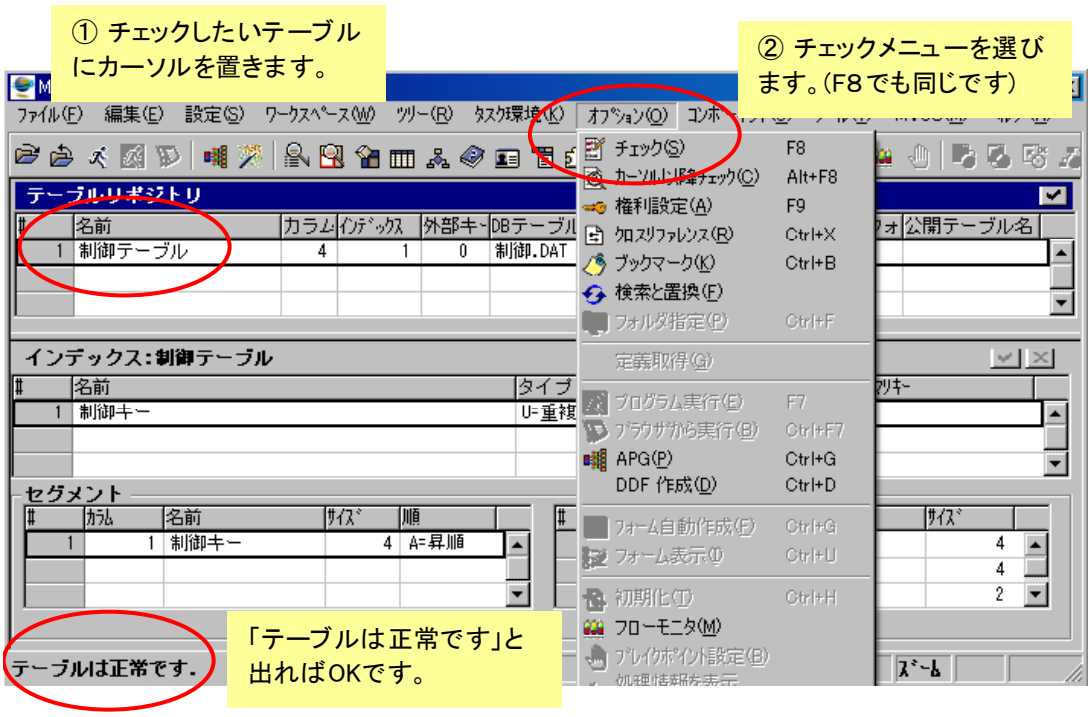


図 4-10 テーブルの文法チェック

もしエラーのあった場合には、エラーメッセージが「チェック結果」ウィンドウに表示されます。エラーテキストをクリックすると、エラーの見つかった箇所に移動するので、適宜修正してください。
図 4-11 は、カラムを空白のままにしてしまっていた場合のエラーの例です。

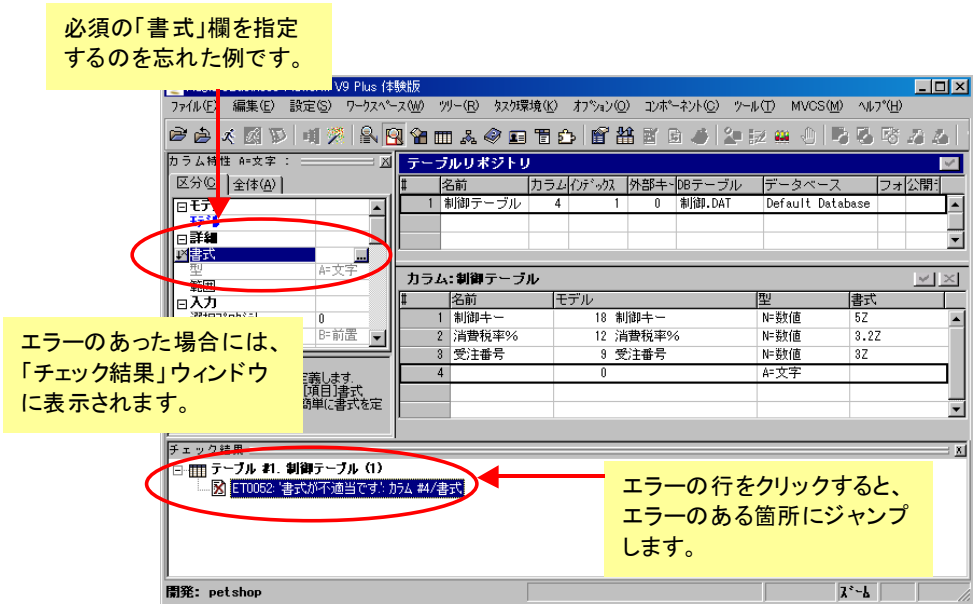


図 4-11 文法エラーのあった例

以上で、制御テーブルについての定義は完了です。

4.4.3. 顧客マスタ テーブル

顧客マスタ テーブルは、顧客に関する情報を格納するマスタテーブルで、カラム定義とインデックス定義は以下の通りです。

表 4-7 顧客マスタテーブルの定義

名前：	顧客マスタ
DB テーブル名：	顧客.MST

顧客マスタ カラム定義					
#	カラム名	モデル番号	モデル名	データ型	書式
1	顧客番号	1	顧客番号	数値	5Z
2	顧客名	2	顧客名	文字	20
3	住所	3	住所	文字	40
4	割引率	13	その他%	数値	N3.2Z
5	条件	4	条件	文字	20
6	受注累計額	16	金額(8 桁)	数値	N8CZ
7	取引回数	15	注文／取引回数	数値	N5CZ
8	顧客メモ	5	顧客メモ	メモ	200

顧客マスタ インデックス定義			
#	インデックス名	タイプ	セグメント
1	顧客番号	重複不可	1 顧客番号
2	顧客名	重複不可	2 顧客名
			1 顧客番号

このテーブルも、制御テーブルのやり方にならって、テーブルリポジトリに定義してみてください。

参考： 「顧客名」インデックスは、顧客名と顧客番号の二つのカラムからなっていますので、セグメントテーブルで図 4-12 のように、二つの行を定義してください。

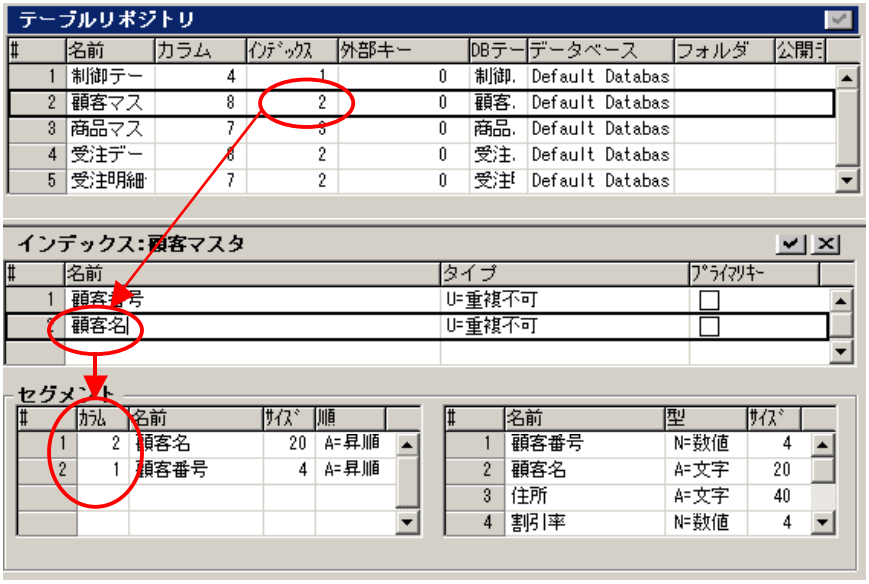


図 4-12 二つのカラム(セグメント)からなるインデックスの定義

4.4.4. 商品マスタ テーブル

商品マスタ テーブルは、商品に関する情報を格納するマスタ テーブルで、以下のカラム定義とインデックス定義とからなります。これも同様にテーブルリポジトリに定義してください。

表 4-8 商品マスタテーブルの基本定義

名前：	商品マスタ
DB テーブル名：	商品.MST

商品マスタ カラム定義					
#	カラム名	モデル番号	モデル名	データ型	書式
1	商品番号	6	商品番号	数値	5Z
2	商品名	7	商品名	文字	20
3	商品タイプ	8	商品タイプ	文字	UA
4	単価	16	金額(8 桁)	数値	N7CZ
5	在庫数	14	商品個数	数値	N5CZ
6	受注数	14	商品個数	数値	N5CZ
7	発注数	14	商品個数	数値	N5CZ

商品マスタ インデックス定義			
#	インデックス名	タイプ	セグメント
1	商品番号	重複不可	1 商品番号
2	商品名	重複不可	2 商品名
			1 商品番号

3	商品タイプ	重複不可	3 商品タイプ
			1 商品番号

4.4.5. 受注データ テーブル

受注データテーブルは、受注１件について１レコードが対応し、各注文に関する情報を格納します。

表 4-9 受注データテーブルの定義

名前：	受注データ
DB テーブル名：	受注.DAT

受注データ カラム定義					
#	カラム名	モデル番号	モデル名	データ型	書式
1	受注番号	9	受注番号	数値	3Z
2	顧客番号	1	顧客番号	数値	5Z
3	最終明細番号	10	受注明細番号	数値	3Z
4	受注日	11	受注日	日付	YYYY/MM/DD
5	明細合計額	16	金額(8 桁)	数値	N8CZ
6	受注割引額	16	金額(8 桁)	数値	N8CZ
7	消費税額	16	金額(8 桁)	数値	N8CZ
8	受注合計額	16	金額(8 桁)	数値	N8CZ

受注データ キー定義			
#	インデックス名	タイプ	セグメント
1	受注番号	重複不可	1 受注番号
2	顧客番号	重複不可	2 顧客番号
			1 受注番号

4.4.6. 受注明細データ テーブル

受注明細データテーブルは、各注文中の明細行の情報を格納し、受注番号と受注明細番号とを組み合わせてキーとします。

表 4-10 受注明細データテーブルの定義

名前：	受注明細データ
DB テーブル名：	受注明細.DAT

受注明細データ カラム定義					
#	カラム名	モデル番号	モデル名	データ型	書式
1	受注番号	9	受注番号	数値	3Z
2	明細番号	10	受注明細番号	数値	3Z

3	商品番号	6	商品番号	数値	5Z
4	商品タイプ	8	商品タイプ	文字	UA
5	数量	14	商品個数	数値	N5CZ
6	単価	16	金額(8 桁)	数値	N7CZ
7	合計	16	金額(8 桁)	数値	N8CZ

受注明細データ キー定義			
#	インデックス名	タイプ	セグメント
1	受注番号	重複不可	1 受注番号
			2 明細番号
2	商品番号	重複不可	3 商品番号
			1 受注番号
			2 明細番号

4.5. APG 実行

テーブルを定義したら、テーブルのレコードを作成しましょう。

テーブルを開いて、レコードを照会・作成・修正・削除するというのは、アプリケーション開発時に頻繁に使う基本的な操作ですが、Magic では **APG** (Automatic Program Generator) という機能を提供しており、APG によりごく簡単にレコードの操作を行うことができます。

4.5.1. APG の実行の方法

以下に、例として顧客マスタをテーブル形式で表示させる場合のやりかたを説明します。

1. 顧客マスタにカーソルを置きます。
2. メニューで「オプション(O)」 → 「APG (G)」を選びます。メニューを選ぶ代わりに、Ctrl+G を押して APG を呼び出すこともできます。

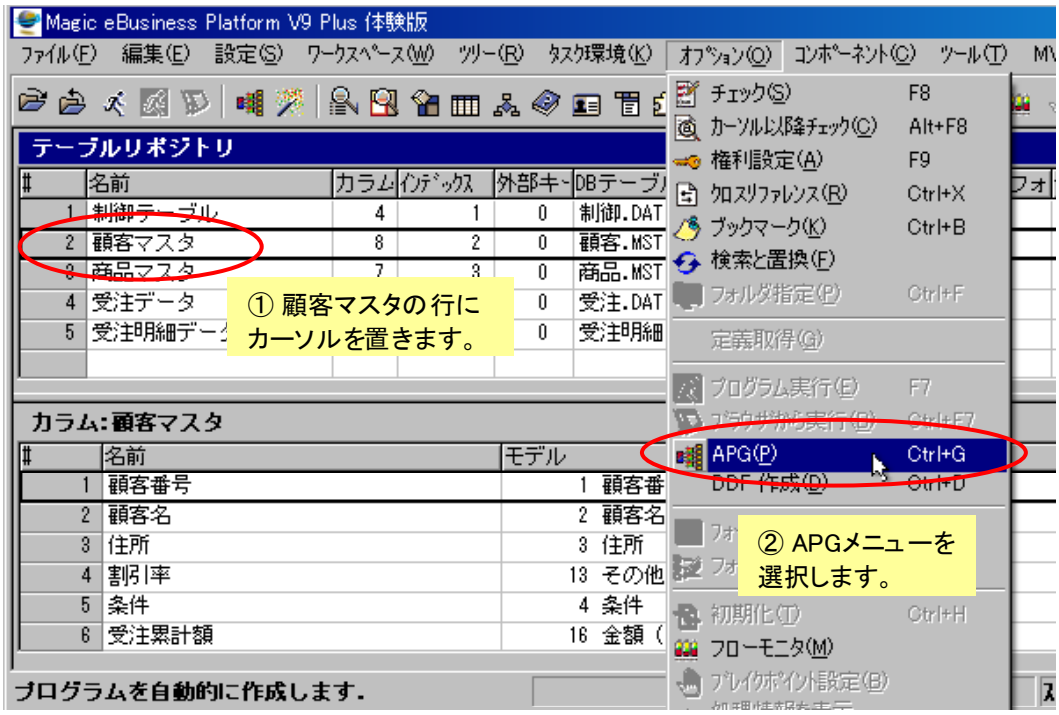


図 4-13 APG の起動

3. 「APG: 顧客マスタ」ダイアログが開くので、そのまま Enter を押してください。

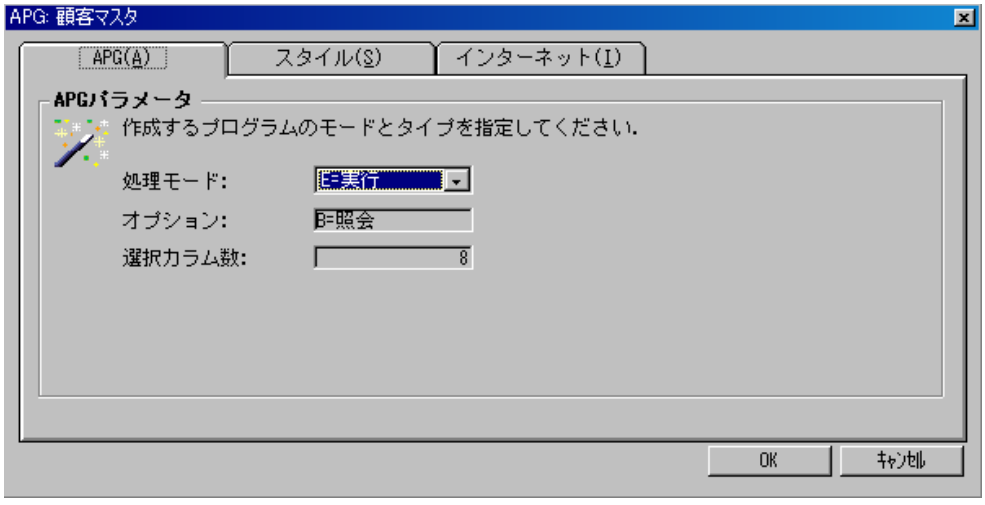


図 4-14 APG ダイアログ

4. 空のテーブルが表示されますので、適当にデータを入力してください。

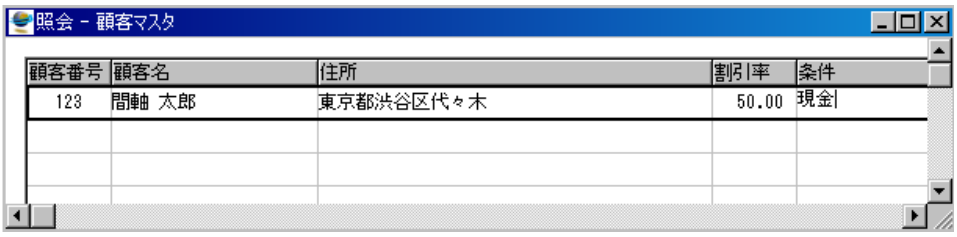


図 4-15 顧客マスタテーブル

5. ESC を押すか、ウィンドウ右上の X ボタンを押すと、終了します。

このテーブルが、Windows 上ではどのように作られているのかを確認してみましょう。エクスプローラで、Magic のインストールされているディレクトリを開いてください。

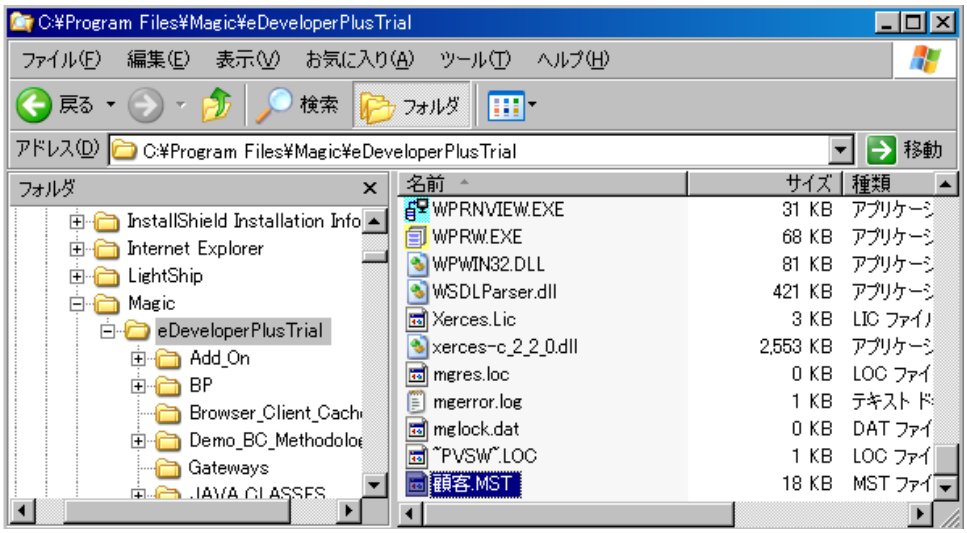


図 4-16 顧客マスタファイル 顧客.MST

ここで見られるように、「顧客.MST」という名前でファイルができていることを確認してください。これが顧客テーブルの実体となるファイルです。

参考： このファイルは、Pervasive SQL のファイル形式して格納されているので、テキストエディッタなどではレコードの内容を見ることができません。

4.5.2. APG のオプション

上の例では、テーブルの内容を 2 次元のテーブル形式で表示させましたが、APG にはそのほかにもいくつかのオプションがあります。

ラインモードとスクリーンモード： APG を起動し、「APG: 顧客マスタ」ダイアログを開いたところで、「スタイル(S)」タブを選び、「表示モード」を「S=スクリーン」にしてみてください。レコードは 2 次元のテーブル形式ではなく、1 レコードづつ表示される「スクリーンモード」で表示されます。

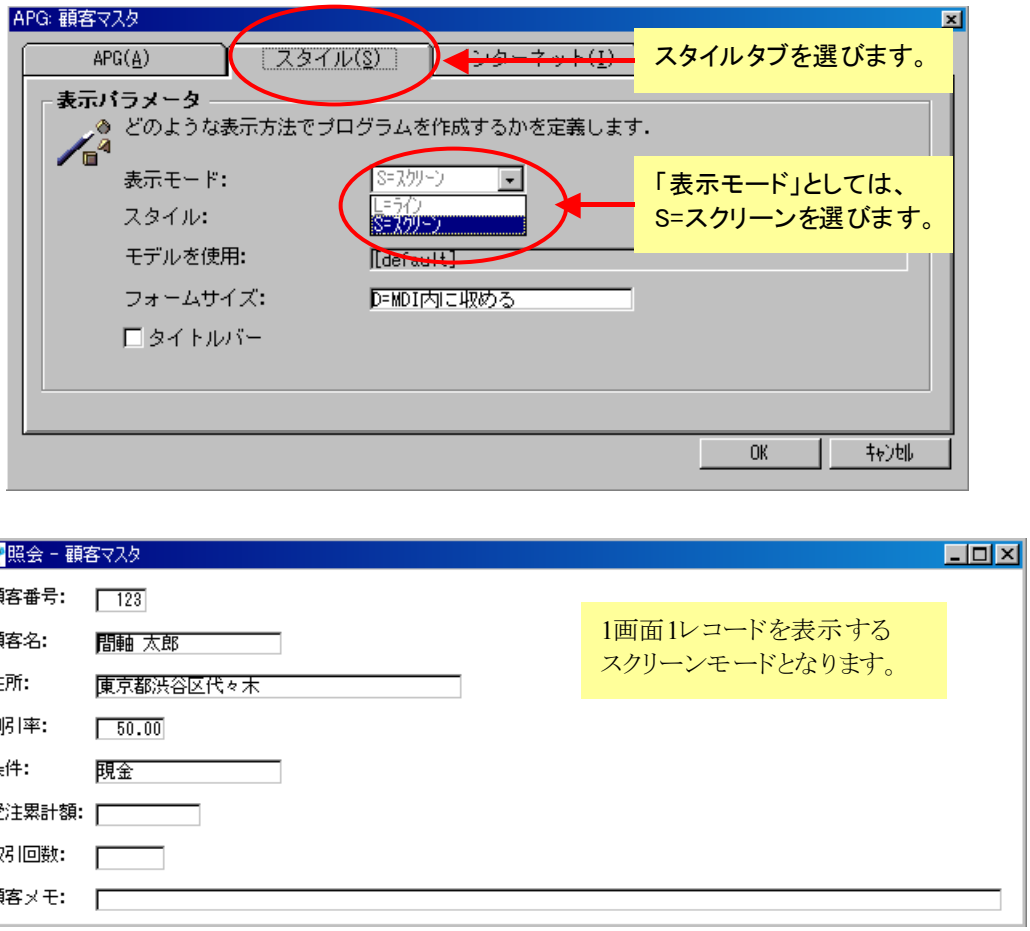


図 4-17 スクリーンモードの APG

プログラムの作成： 今までは、APG ですぐにデータ内容を表示していました。このときに内部では、テーブル表示用の一時プログラムが作成され、実行され、終了したら削除される、という処理が行われています。APG ではこの他に、表示用プログラムを作成してテーブルリポジトリに登録するだけで、すぐには実行しない、というオプションもあります。

1. 「APG: 顧客マスタ」ダイアログで、「処理モード」を「G=作成」とします。

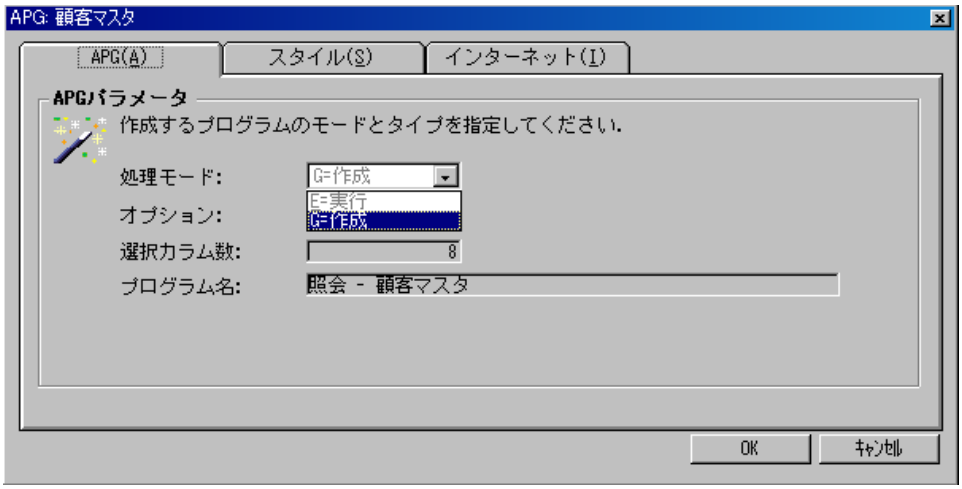


図 4-18 プログラムの作成モードの APG

2. プログラムリポジトリを開いてみてください。ツールバーから開くことができます。

図 4-19 プログラムリポジトリをツールバーから開く

3. プログラム「照会 – 顧客マスタ」が作成されています。

図 4-20 プログラムリポジトリに登録された表示プログラム

4. ここで F7 を押してみてください。プログラムが実行されます。

図 4-21 プログラムリポジトリからプログラム実行

4.6. ファイルを用意する

レコードをひとつづつ入力していくのは大変なので、デモでは予めデータを作成したものを用意してありますので、以後はこれを使います。

1. Magic をインストールしたディレクトリの下にある、Petshop¥Data ディレクトリに移動します。以下のような五つのファイルがあります。

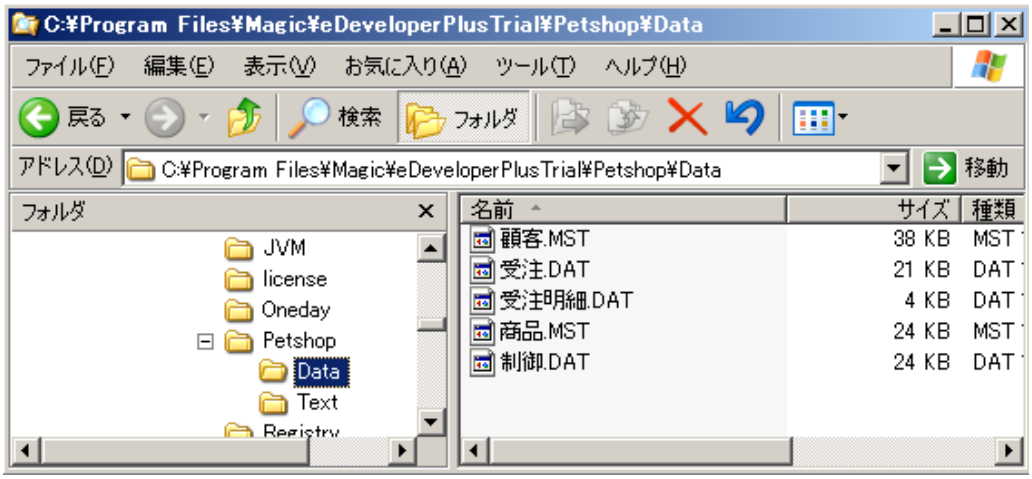


図 4-22 ペットショップデモのデータ

2. このファイルをすべて選択し、Magic をインストールしたディレクトリに上書きコピーします。単純なファイルコピーですので、以下のような手順で行えます。
   (ア) Ctrl + A ですべてを選択、
   (イ) 右マウスクリックメニューで「コピー」、
   (ウ) Magic をインストールしたディレクトリに移って、右マウスクリックメニューで「貼り付け」

確認のために、テーブルリポジトリで APG を行い、テーブルの内容を確認してください。

顧客番号	顧客名	住所	割引率	条件	受注累計額	取引回数
1008	千葉ペットショップ	千葉県千葉市高柳　１２３４－１	9.00	３０日後支払い		
1234	ペットセンター神田	東京都千代田区神田　１－２－３	10.00	現金		
3201	コジマペット	東京都足立区綾瀬 ３－１１－５	5.00	現金		
3220	ペットショップワンワン	東京都江戸川区南篠崎町 ３－３２２	10.00	３０日後支払い		
3321	ヤザキ金魚	東京都杉並区高井戸東 ３－５－６	5.00	３０日後支払い		
3440	ペットサロンヤザワ	東京都文京区小石川 ２－５－７	3.00	現金		
3550	ペットワールドハート	愛知県名古屋市名東区高針 ３６５２	8.00	４５日後支払い		
4920	ANIMAL HOUSE	京都府京都市伏見区醍醐東大路町２３	5.00	現金		
5133	山田ペットハウス	岡山県岡山市表町３－６－９	3.00	現金		
5387	フクトミ鳥の友	佐賀県唐津市和多田本村５－５	5.00	現金		
5493	わんわんペット	北海道札幌市厚別区厚別西３条２２－３	3.00	４５日後支払い		
5678	サンシャインペット	東京都豊島区池袋　４－１－１２	15.00	４５日後支払い		
6238	アラジンアニマル	宮城県仙台市青葉区川平１－５－３	5.00	現金		
6588	ペットハウスシャボン	秋田県秋田市東通３－１１－２２	4.00	現金		
6688	犬猫ブラザーズ	山形県山形市平久保１－３５	2.00	現金		
7834	酒田兵衛門錦鯉園	福島県郡山市安積町成田２９－３	5.00	現金		
7963	(有)トンビ鷹	茨城県鹿嶋市高天が原３－５－１	3.00	現金		
8135	キューダブルイー	埼玉県浦和市常盤５－６－７	7.00	現金		
8256	春日部ガーデン	埼玉県春日部市八丁目１００－１	4.00	現金		

図 4-23 顧客マスタデータの内容

参考：

- 後でデータを初期化したくなった場合にも、同じ手順でファイルを上書きコピーすれば、データは最初の状態に戻ります。

5. スクリーンモードオンラインプログラム

データ定義ができたので、Magic プログラムを作っていきます。
最初は Magic のプログラムの一番基本的な形である、オンラインデータ照会プログラムを作ります。例としては、スクリーンモード(1 画面 1 レコードで表示する形式) で制御テーブルを表示するプログラムを作成します。

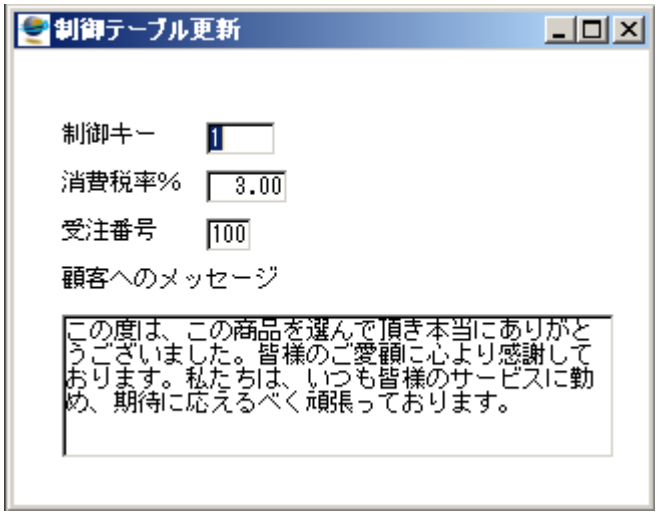


図 5-1 制御テーブル更新プログラム(最終形)

参考情報： 本章の内容については、リファレンスマニュアルにより詳細な情報がありますので参考にしてください。

APG	第 19 章 ユーティリティ → APG (自動プログラムジェネレータ)
プログラムリポジトリ全般	第 6 章 プログラム
タスク特性	第 6 章 プログラム → タスク → [タスク特性]ダイアログ
レコードメインなどの処理レベル	第 5 章 Magic アプリケーションエンジン → エンジンレベル 第 6 章 プログラム → タスク定義ウィンドウ
フォーム	第 6 章 プログラム → フォームテーブル
フォームの扱い	第 9 章 表示フォーム → GUI 表示フォーム、GUI 表示コントロール、項目パレット、GUI 表示コントロール特性など
Magic 実行エンジンの詳細	第 5 章 Magic アプリケーションエンジン
オンラインプログラムの実行フロー	第 5 章 Magic アプリケーションエンジン → レコードループのフローチャート

5.1. APG により作成する

オンラインの表示プログラムを作成するには、APG を使うのが一番簡単です。前章で顧客マスタの照会プログラムを、テーブルリポジトリから APG で作成しましたが、今度は制御テーブルをスクリーンモードで表示するプログラムを、プログラムリポジトリから作成してみましょう。

5.1.1. プログラムリポジトリからの APG 実行

1. Shift+F3 でプログラムリポジトリを開きます。
   「メインプログラム」と、先ほど作成した「照会 – 顧客マスタ」プログラムがあります。
2. 「顧客マスタ」にカーソルを移動し、F4 で新規プログラムを作成します。
3. Ctrl+G で APG を起動します。
4. 「メインテーブル」欄からズームして、「制御テーブル」を選びます。

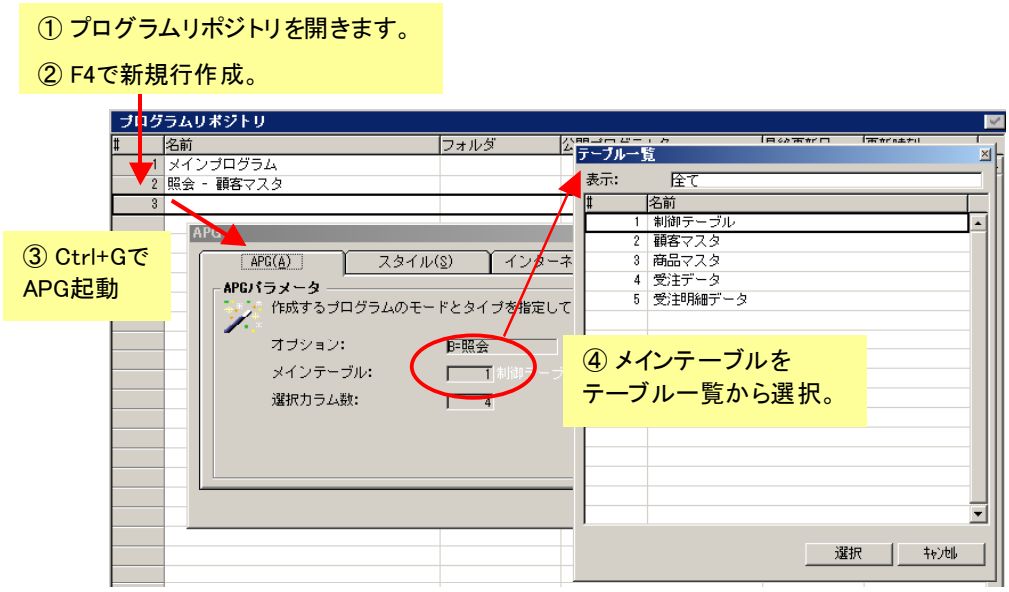


図 5-2 APG で表示プログラムを作成（1）

5. 「スタイル」タブを開き、「表示モード」を「S=スクリーン」にします。
6. OK ボタンを押します。
   「照会 - 制御テーブル」プログラムが作成されます。

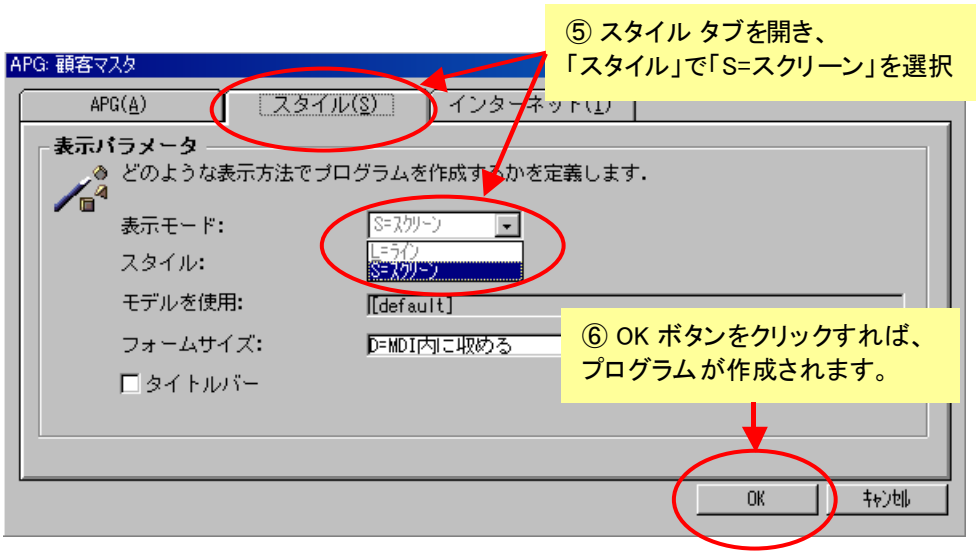


図 5-3 APG で表示プログラムを作成 (2)

5.1.2. 実行してみる

作ったプログラムを実行してみます。

プログラムを実行させるには、次のようにします(図 5-4)。

1. 実行させたいプログラム「照会 - 制御テーブル」にカーソルを置きます。
2. 実行に先立って、文法チェックを必ず行いましょう。チェックはテーブルリポジトリの場合と同じく、F8 キーを押します。正常ならば「プログラムは正常です」とステータスバーで表示されます。エラーのある場合には、「チェック結果」ウィンドウにエラーメッセージが出ますから、修正してください。
3. F7 キーを押します。プログラムが実行されます。

参考：

プログラムの実行には、F7 キーのほかに、次のような方法もあります。

(ア) ツールバーで「プログラム実行(E)」アイコンを押す。

(イ) メニュー 「オプション(O)」→「プログラム実行(E)」を選ぶ。

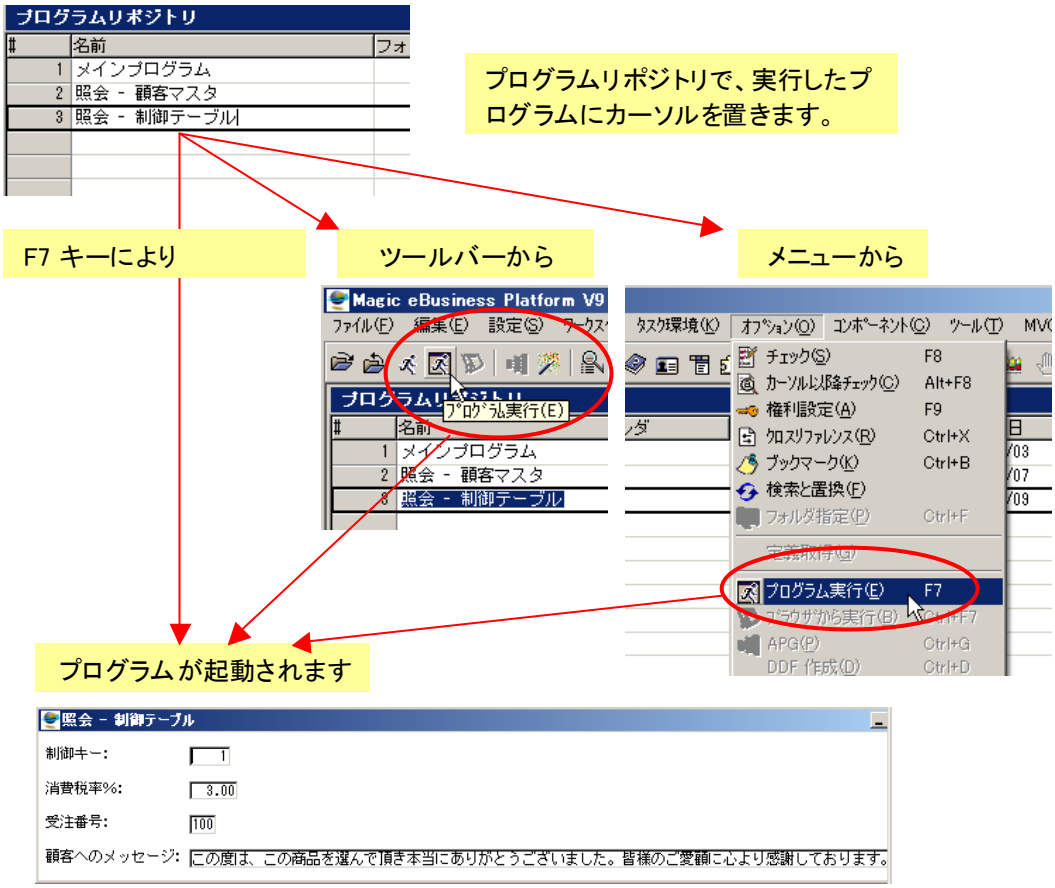


図 5-4 プログラムリポジトリからプログラム実行

4. プログラムを終了するには、ウィンドウ右上の X ボタンを押すか、あるいは ESC キーを押します。

5.2. APG を使わずに作成する

APG を使う方法は簡単ですが、勉強のため、APG を使わずに、ゼロから同じプログラムを作ってみましょう。

5.2.1. タスク特性の設定

タスク特性は、プログラムの基本的な性質を定義するダイアログで、次のようなパラメータをここでは設定します。

- タスク名（全角や特殊文字など含め、任意の文字が使えます)
- タスクのタイプ（ユーザの入力があるオンラインタスク、ユーザの介入なしで実行を行うバッチタスクなどがあります)
- 初期モード（表示のみの**照会モード**、データの修正が可能な**修正モード**、新規データ登録のための**登録モード**などがあります)
- メインテーブル（表示を行いたいテーブル)
- キー（レコードの表示順を、テーブルに定義されたインデックスにより指定します)。

Magic の開発環境では、新しくプログラムを作った場合には、必ず最初にこのタスク特性を定義します。以下の要領でタスク特性を定義してください。

1. プログラムリポジトリで F4 を押し、新規プログラムを作成します。
2. F5 を押します。プログラムが開き、「プログラム特性」ダイアログが開きます。
3. 名前は「制御テーブル更新」、タスクタイプは「O=オンライン」、初期モードは「M=修正」（データの修正が可能）と設定します。
4. メインテーブル欄からズームし、テーブル一覧から「制御テーブル」を選びます。
   参考： テーブルの番号を直接入力することもできます。
5. インデックスには 1 番（「制御キー」インデックス)が自動的に設定されるので、そのままにします。
6. OK ボタンをクリックします。

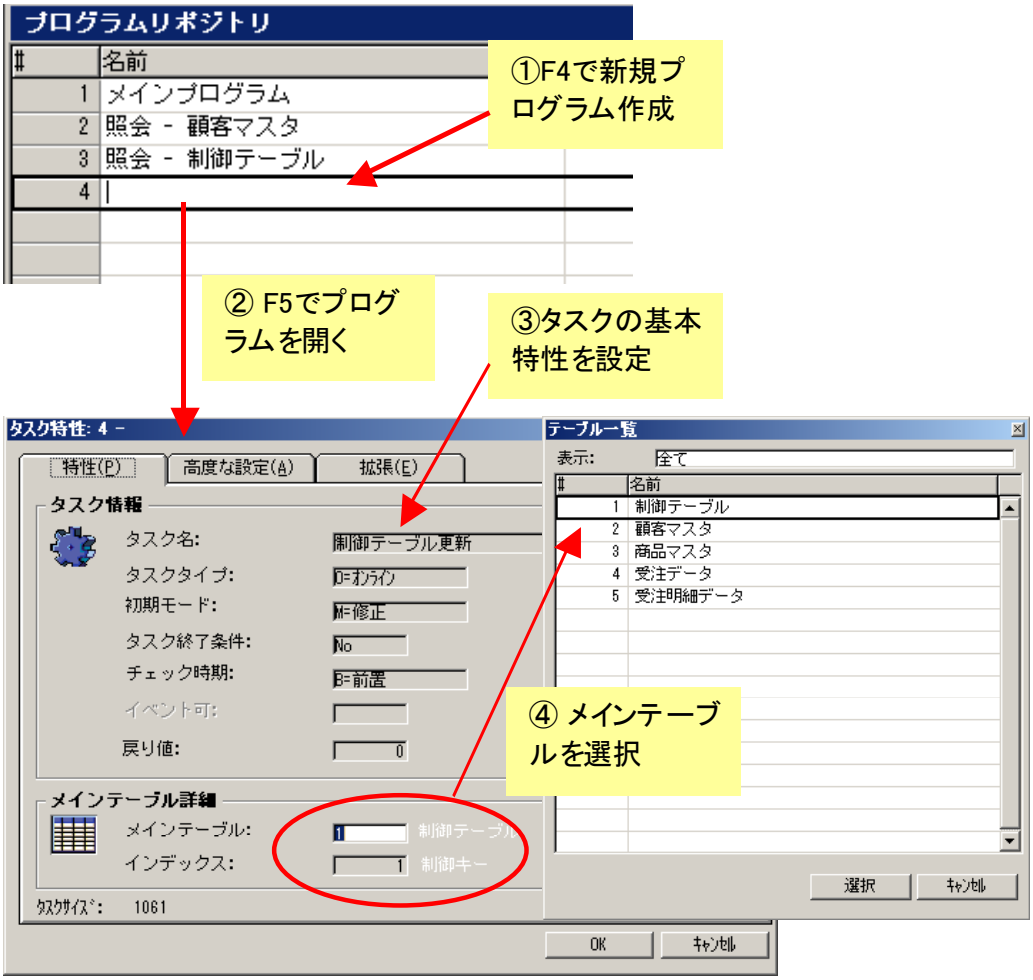


図 5-5 タスク特性の定義

5.2.2. レコードメインの定義

タスク特性を定義したら、次に行うのは、このタスクで利用するデータ項目を定義することです。
Magic で定義できるデータ項目には、次の 3 種類があります。

- **変数データ項目**： これは通常のプログラム言語などの「変数」と同じもので、プログラムが開始されたときにメインメモリ上に領域が割り当てられ、プログラムが終了したときには解放されます。
- **実データ項目**: 変数データと似ていますが、テーブルのカラムと連動しているところが異なります。実データ項目の初期値はデータベースに格納されているテーブルのカラムから取得されます。ユーザが実データ項目のデータを変更したりした場合には、変更した値が自動的にデータベースに書き戻されます。
- **パラメータ項目**： 変数データ項目と同じですが、別プログラムがこのプログラムを呼び出す際、パラメータとして値を受け渡しすることができるものです。VB や C でも関数やプロシージャのパラメータというものがありますが、それと同じです。

今作ろうとしている制御テーブル更新プログラムでは、制御テーブルに格納されているデータを表示するだけですので、実データ項目だけを利用することになります。変数やパラメータを利用する例は後の章で出てきます。

制御テーブル更新プログラムでは、制御テーブルの全カラム(制御キー、消費税率%、最終受注番号、顧客へのメッセージ)を利用しますので、これを全部選択します。データ項目の定義には「セレクト」コマンドを用います。

1. 「タスク定義」テーブルで、「R=レコード」「M=メイン」の行にカーソルを合わせます。
2. 「処理テーブル： レコードメイン」テーブル上をクリックします。
   カーソルが処理テーブルのタイトル行に移ります。
3. F4 で新規行を作成します。
4. 行の左隅をダブルクリックするとコンボボックスが現れるので、その中から「S=セレクト」を選びます。
   参考： コンボボックスから選択する代わりに、直接 S と入力しても同じです。
5. Tab キーを押すと「R=実データ」のところに移りますが、ここでは実データ項目を選択するので、そのままにしておきます。
6. Tab キーを押すと「0」のところに移りますが、ここから F5 でズームして、カラム 1 を選択します。
   「内容」欄には、カラムの名前が自動的に設定されます。
   参考： カラム番号を直接入力することもできます。

これで、「制御キー」が実データ項目として選択され、このタスクの中で利用できるようになりました。

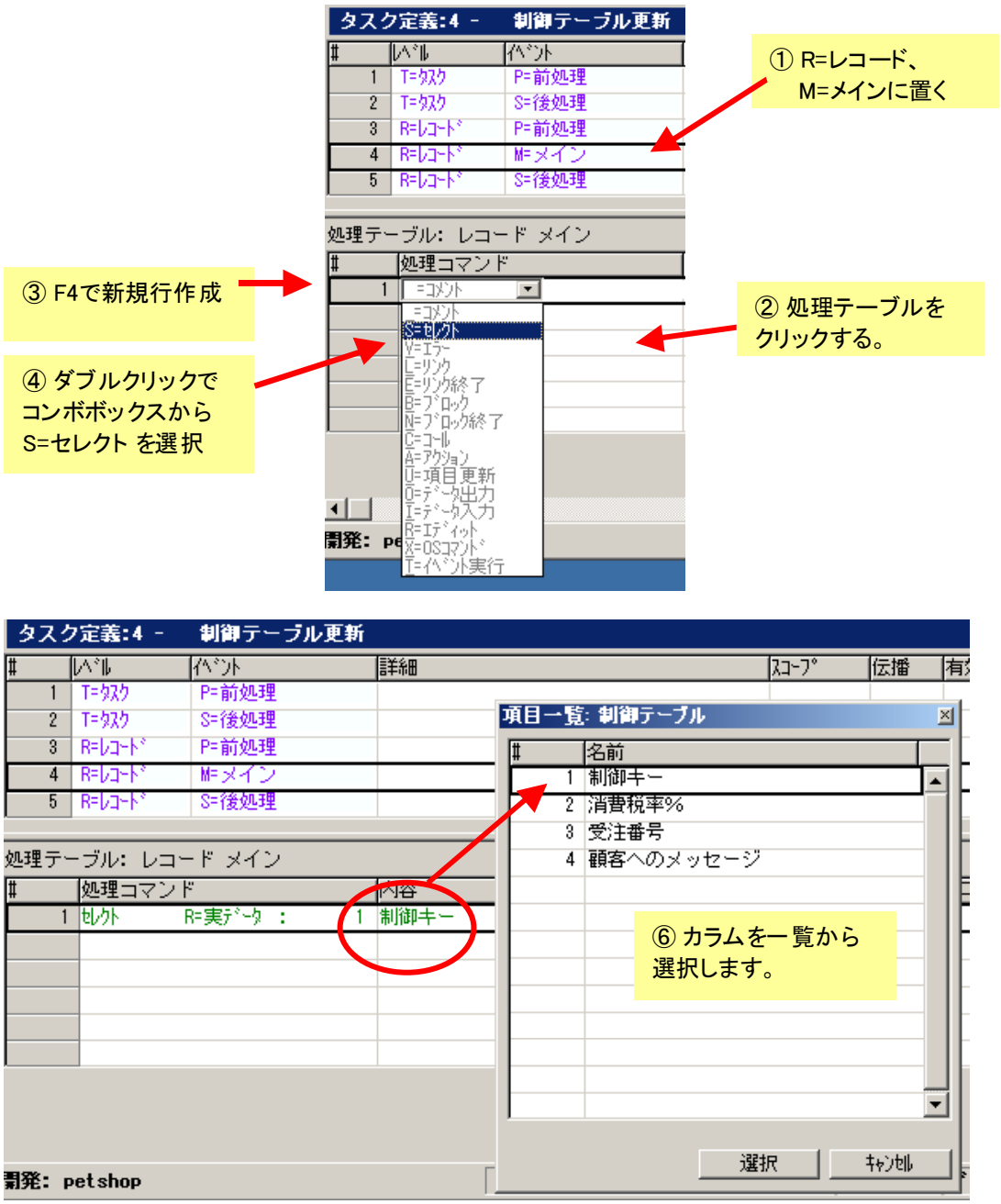


図 5-6 実データ項目の選択

同様にして、全カラムをセレクトコマンドで選択してください。最終的には次の図のようになります。

処理テーブル: レコード メイン

#	処理コマンド			内容			範囲		位置付	
1	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ	: 1	制御キー	代入:	0	0	0	0	0
2	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ	: 2	消費税率%	代入:	0	0	0	0	0
3	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ	: 3	受注番号	代入:	0	0	0	0	0
4	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ	: 4	顧客へのメッセ~	代入:	0	0	0	0	0

図 5-7 制御テーブル更新プログラムのレコードメイン

5.2.3. フォームの作成

プログラムで使うデータ項目を定義したら、次は、ユーザにデータを表示するための**フォーム**を定義します。

1. メニュー「タスク環境(K)」→「フォーム(F)」を選びます。フォームテーブルが開きます。

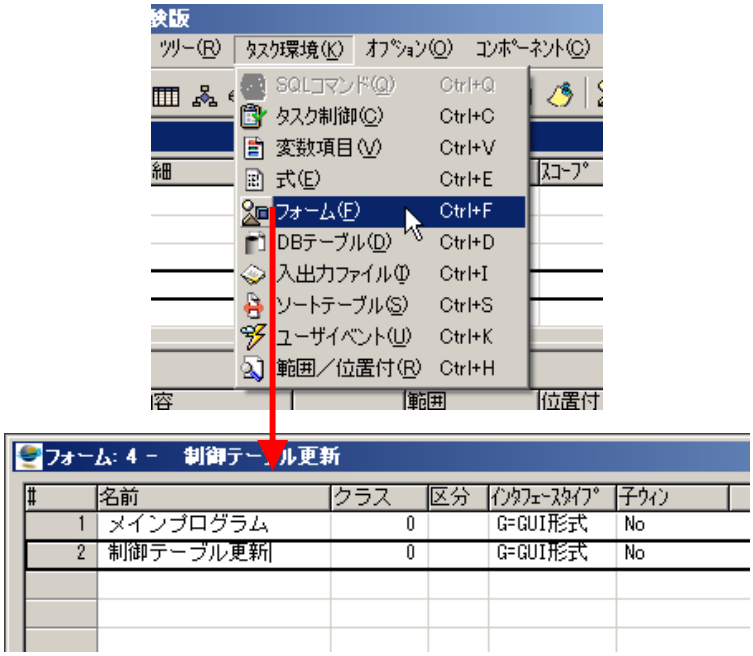


図 5-8 フォームテーブルの表示

2. 「制御テーブル更新」の行で F5 を押します。
   空のフォームが表示されます。
3. コントロールパレットで「項目」タブをクリックします。
   項目一覧が表示されます。
4. 先頭の項目 A をクリックすると、マウスの形状が変わります。このままフォーム上の適当な位置でマウスクリックすると、項目 A がエディットコントロールとしてフォームに貼り付けられます。

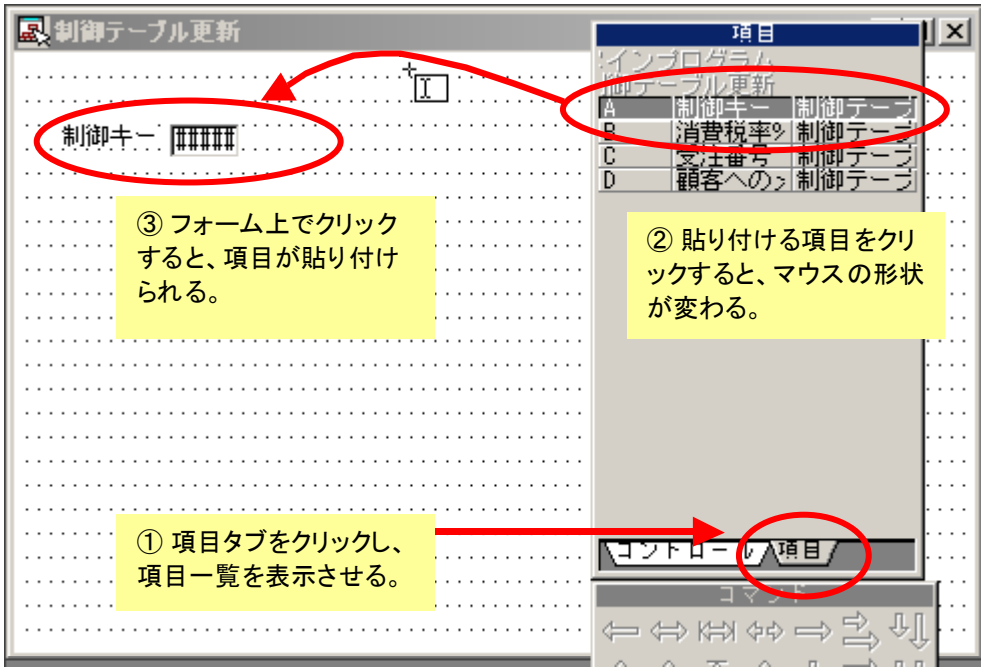


図 5-9 項目のフォームへの貼り付け

5. 同様に、項目 B ～ D を貼り付けます。
6. コントロールやフォームのサイズや位置は、マウスドラッグで調整できるので、きれいに並べてください。
7. 「顧客へのメッセージ」は、長さが 200 バイトあるので、フォーム中に収まりません。この項目は、複数行に表示させるようにしましょう。
   コントロールを選択し、「コントロール特性」の「複数行編集」を Yes にします。
   その後、コントロールの高さを変えて、4 ～ 5 行表示できるようにしてください。

最終的には、次のようなフォームになります。

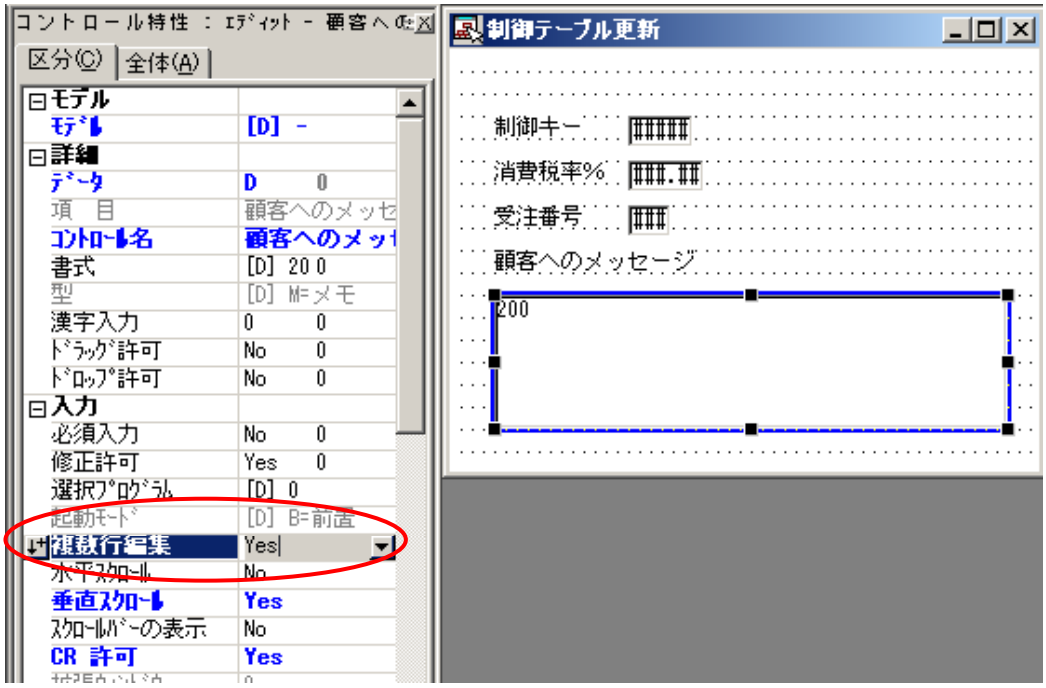


図 5-10 制御テーブル更新プログラムのフォーム

フォームエディッタを抜けるには、ESC キーを押します。あるいは、ウィンドウ右上の X ボタンをクリックしても同じです。

以上でプログラム作成は終了です。

プログラムを閉じるには、Enter キーを何回か押します。プログラムリポジトリにまで戻ってください。

補足：

上ではひとつひとつの項目をフォームテーブルに貼り付けていましたが、フォームテーブル上で、フォーム APG を行ってフォームを自動作成することもできます。

1. フォームテーブルを開く。
2. Ctrl+G を押す。→ 「変更しますか？」の確認ダイアログが出るので、「はい」を押します。現在フォームに定義されているものはすべて消去されてしまいます。
3. 「作成フォーム」ダイアログが出るので、「表示モード」を「S=スクリーン」に直し、Enter を押します。これで、レコードメインにセレクトされている項目がすべて、スクリーンモードで貼り付けられます。
4. F5 でフォームエディッタを開いて、自動作成されたフォームの内容を確認してください。

5.2.4. 実行してみる

今作ったプログラムを実行してみましょう。

1. F8 (プログラムチェック)でチェックします。「プログラムは正常です」がステータスバーに表示されることを確認してください。万一エラーがあった場合には、エラーメッセージに従って修正してください。
2. F7 キーを押します。

プログラムが実行され、データが表示されます。

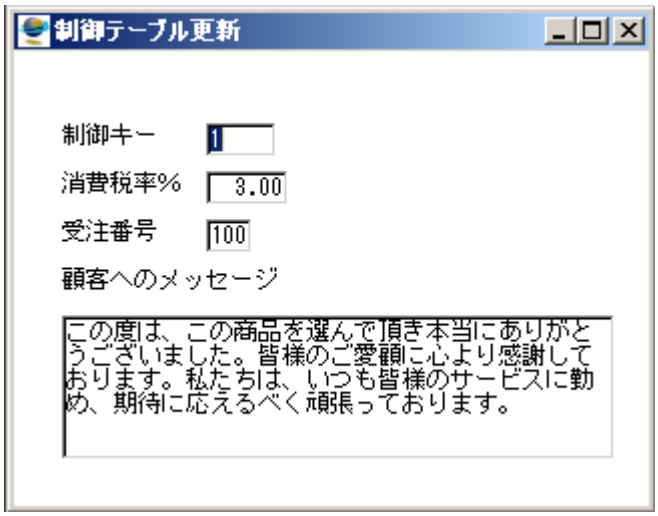


図 5-11 制御テーブル更新プログラム実行画面

ここで、いくつか簡単な事柄を確認しておきましょう。

- **データの修正：** 「顧客へのメッセージ」の文面を変更して、ESC キーで一旦終了し、再度 F7 キーで実行してください。修正した内容が反映されていることを確認してください。
  これにより、実データ項目は、データベース内のデータと連動しており、レコードメインで「セレクト」コマンドで定義するだけで、データ修正内容が自動的にデータベースに反映されることが確認できます。
- **修正の取り消し：** 同じく、「顧客へのメッセージ」の文面を変更して、今度は ESC キーで終了せず、F2 キーを押します。修正した内容が取り消され、最初のデータに戻されていることを確認してください。

5.3. 裏で何が起こっているのか？

以上見たように、Magic ではデータベースからのレコードの読み込みや書き込みなどは、開発者がプログラムせずともすべて自動的に行われることがわかると思います。では Magic の実行時には、裏で何が起こっているのでしょうか？

Magic のプログラムは、VB や C などでのプログラミングとは異なり、プログラムについての仕様を定義していくという、非常にレベルの高い記述方法で作成していきます。これが Magic の高い生産性と保守性、そしてプラットフォームの移植容易性の秘密です。

開発者が作成した Magic のプログラムを実行するのは、Magic の**実行エンジン**です。Magic のプログラムが Magic の実行エンジンで実行されるのは、ちょうど Java や C# などが、JavaVM や CLR などの仮想マシン上で実行されるのに似ています。

しかし Magic の実行エンジンは、汎用的な処理を念頭において設計されている Java や.Net の仮想マシンとは異なり、データベースアプリケーション向けに最適化されて設計されており、データベースアプリケーションでよく見られる定型的な処理をすべて自動的に行う、高度なインターフェースを持ったものです。その正確な内容は高度で複雑ですが、概略は次に説明していきます。

5.3.1. プログラムとタスク

Magic エンジンについて説明するまえに、「プログラム」と「タスク」という言葉について補足しておきます。

プログラムリポジトリで F4 キーを押すと、新しい行が作成されます。これにより新しいプログラムがひとつ作成されたことになります。

F5 キーを押すと、プログラムが開きます。そして、最初に「タスク特性」ダイアログが現れます。また、メニューには「タスク環境」があり、処理テーブルは「タスク定義」と表されています。

では、プログラムとタスクとはどういう関係にあるのでしょうか？

後の章で出てきますが、タスクは複数の「サブタスク」(子タスクとも言う)を持つことができます。サブタスクは親タスクに依存して存在し、親タスクのデータ項目などにアクセスすることができます。これはプログラム言語の関数定義のネストに似ています。

また、サブタスクはさらに複数のサブタスク(一番上のタスクから見たら、孫タスク)を持つことができます。このようにタスクはツリー状の子、孫、ひ孫・・・ を持つことができます。

このようにして作られたツリー状のタスクの全体をひとつの「プログラム」と呼びます。

本章で作った制御テーブル更新プログラムでは、サブタスクを作っていないので、このプログラムはただひとつのタスクだけからなります。したがってこの場合にはプログラム＝タスクとなります。

Magic 実行エンジンの実行の単位はタスクであるので、以後の説明では、「タスク」の実行方式について説明をしていきます。

5.3.2. フォーム、ビュー、データベースの関係

プログラムで使うデータ項目（実、変数、パラメータを問わず)は、レコードメインで「セレクト」コマンドを使って定義されますが、ひとつのタスクで定義されているデータ項目の全体をさして、**ビュー**と呼びます。ここで言うビューは、リレーショナルデータベースで言うところの「ビュー」とは違いますので、混同しないでください。

フォーム、ビュー、データベーステーブルと、データの関連について示すと、次の図のようになります。

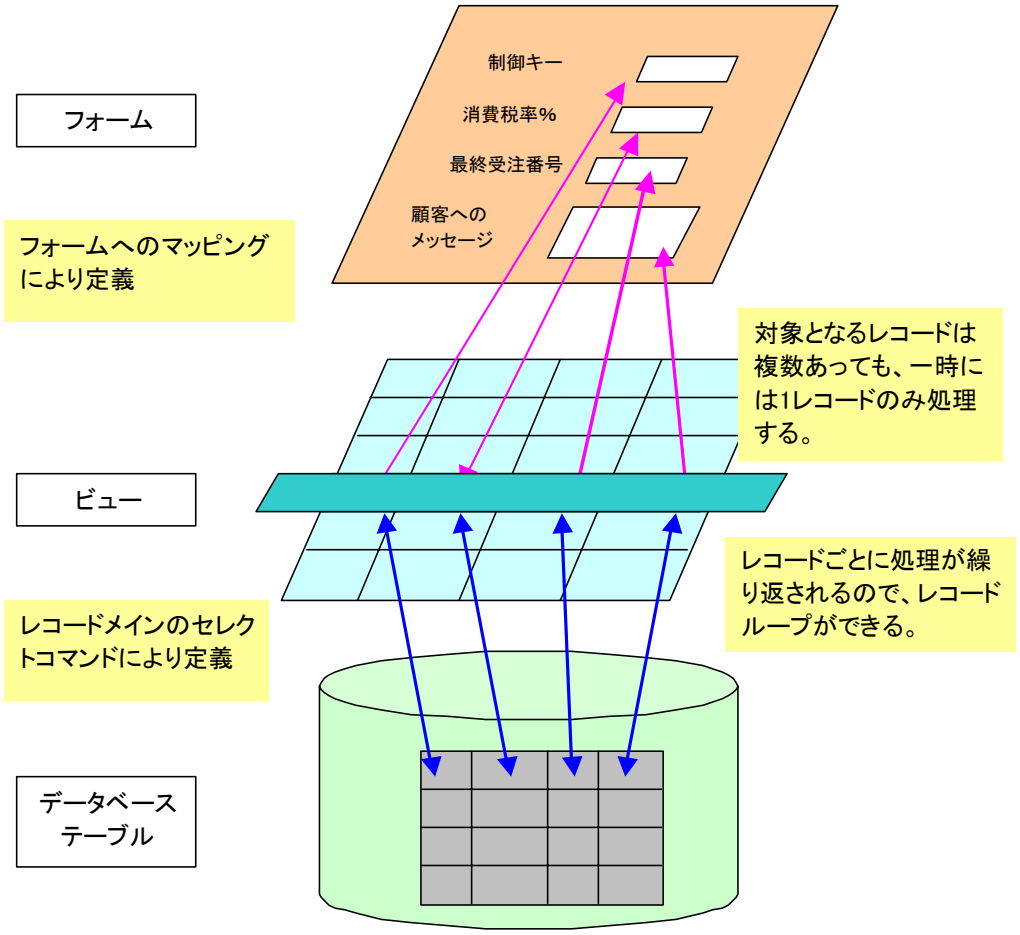


図 5-12 フォーム、ビュー、データテーブルの関連図

- データベースに格納されているテーブルとビューとの関連（青色の矢印で示されたもの)は、レコードメインの「セレクト」コマンドにより定義されます。
  本章の例では、制御テーブルのカラムをすべて選択していましたが、必要がなければ選択しないカラムがあってもかまいません。
- ビューは、セレクトコマンドにより選択されたデータ項目からなります。
  本章の例では、すべて実データ項目のみを使いましたが、データベーステーブルと関連づけられていない、変数データ項目やパラメータ項目などもビューの一部となります。
- ビューには、一時には 1 レコードのデータだけを読み込みます。一般にはデータベーステーブルに複数のレコードがあり、複数のレコードが処理対象となりますが、一時には 1 レコードづつ処理されます。したがって、Magic のプログラムでは、複数のレコード処理のために、ループが行われることになります。これをレコードループと呼びます。
- 対象となるレコードを絞り込むために、「範囲」付けを指定することもできます。これはセレクトコマンドで定義します。
- プログラムでは、処理の主な対象となるメインテーブルからビューを作りますが、ほかのテーブルをリンクしてビューを拡張することもできます。Magic のデータのリンクは、リレーショナルデータベースにおける外部結合（outer join)に似ています。

例えば、受注データレコードには顧客コードのカラムがありますが、顧客名は格納されていません。顧客名もフォームに表示したい場合には、顧客マスターテーブルに対して顧客コードを条件としてリンクを行い、レコードを取り込んできます。この場合には、受注テーブルがメインテーブルであり、顧客マスタはリンクテーブルとなります。一般には、メインテーブルはひとつだけですが、複数のテーブルにリンクをすることができます。

- ビューの実項目が変更された場合には、レコードループの終わり（ひとつのレコードから次のレコードに移る)のタイミングで、データベーステーブルに書き込みが行われます。
- フォームには、ビューの中にあるデータ項目を関連づけます。フォームにはすべてのデータ項目を貼り付ける必要はなく、必要なもののみを貼り付けます。
  表示のみでよい場合には、フォーム上に式で計算した値を表示させることもできます。

5.3.3. 実行のフロー

Magic エンジンの実行のフローを、フローチャート的に書くと、次の図のようになります。これはオンラインプログラムの場合の概略図であり、実際にはもっと複雑で細かな動きがあります。
図中、青色の部分は、Magic エンジンが自動的に裏で実行してくれる部分であり、白色の部分が開発者が定義する部分です。
ここで見るように、データベース処理で必ず出てくる、テーブル(ファイル)のオープン・クローズ、範囲の計算、レコードの読み込みと書き込み、レコードのループなどの定型的で低レベルな処理はすべて Magic エンジンが行ってくれるので、開発者はアプリケーションのロジックに専念することができるようになっています。

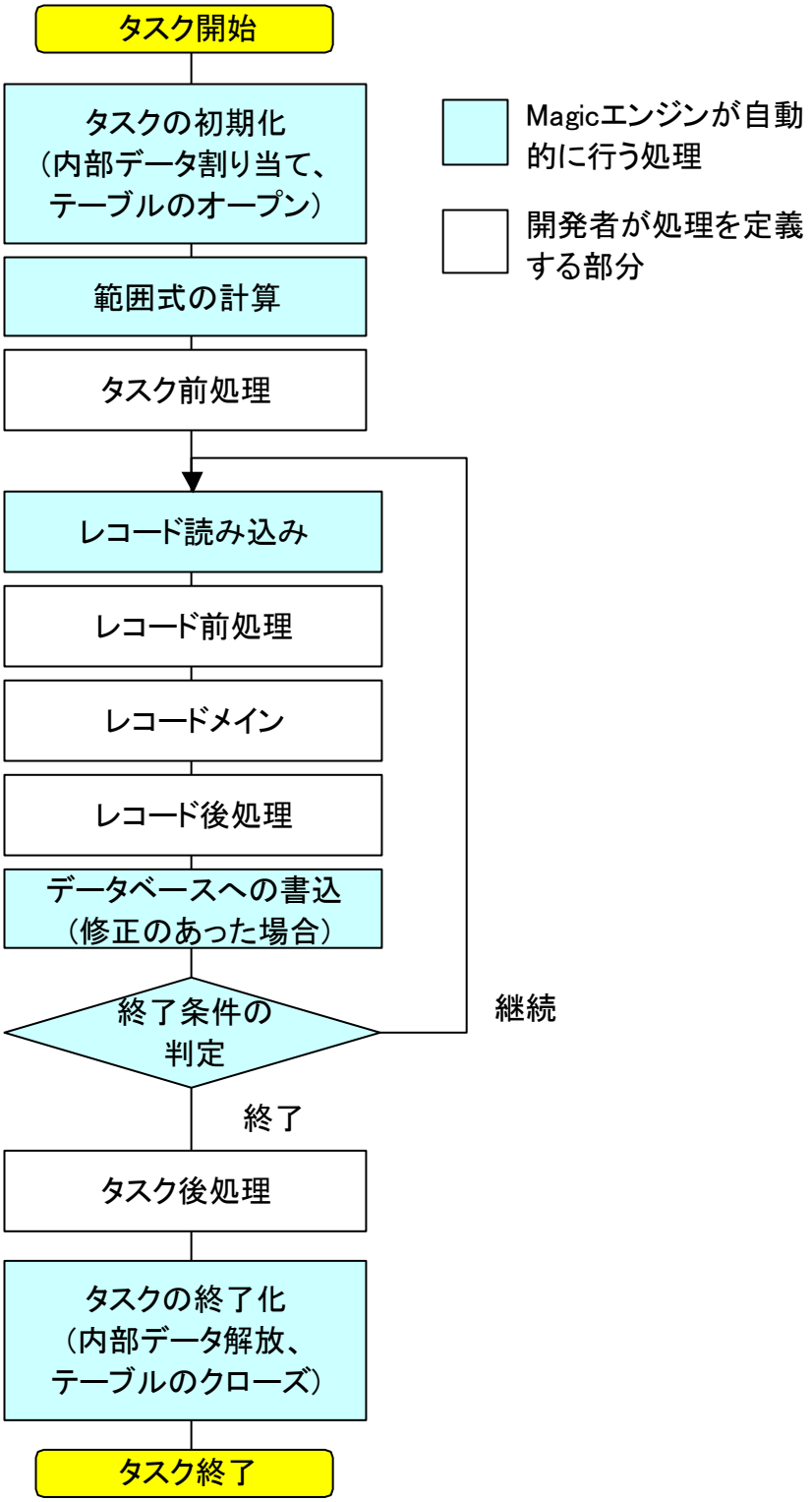


図 5-13Magic 実行エンジンのフロー（オンラインの概略図）

1. タスクの初期化

(ア) タスクのオープン。実行しようとするタスクの定義を MCF ファイルより読み込み、実行に必要な内部データ構造などを準備します。

(イ) タスクで使うテーブル(ファイル)のオープン。上の例では、制御テーブルを使っていましたので、これをオープンします。

2. 範囲式の計算。メインテーブルのレコードの中から、実際の処理対象とするレコードを絞込みたい場合には、セレクトコマンドで範囲式を指定しますが、ここでその式を評価します。
3. タスク前処理： 最初のレコードの処理に入る前に行っておきたい初期化処理などがあればここで行います。
4. レコードレベルの処理
   (ア) 最初のレコードをフェッチ(読み込み)します。読み込んだレコードのデータはメモリ内に格納します。メモリ内では、実データ項目も変数データ項目も同じように扱われます。
   (イ) ビューの初期設定： セレクトコマンドの「代入：」式と、データリンクとがここで行われます。
   (ウ) レコード前処理： 各レコードごとの初期化処理などがあればここで行います。
   (エ) レコードメイン： ビューの内容を、フォーム上のエディットコントロールなどに表示し、ユーザ入力を待ちます。ユーザがフォーム上のデータを修正したら、それに合わせてメモリ上のビューの内容も変更します。この時点では、変更はまだデータベースに反映されていません。
   (オ) ユーザが ↓、↑、PgUp、PgDn、ESC などのキーを押した場合、レコードメインは終了します。
   (カ) レコード後処理： レコードの書き込みを行う前に行っておきたい処理があれば、ここで行います。これは、ビューに変更があった場合にのみ実行されます。
   (キ) データベースへの書き込み： 実データ項目に変更があった場合には、修正データがデータベースに書き戻されます。
   (ク) 終了条件の判定： ESC をした場合には、終了の方に行きます。その他の場合には、別レコードについて、レコードのループが始まります。
5. タスク後処理： タスクの最後に行いたい処理があれば、ここで行います。
6. プログラムの終了化処理
   (ア) テーブルをクローズします。
   (イ) プログラムで使っていた内部データなどを解放し、プログラムを終了します。

以上、概略ではありますが、Magic エンジンの動作を説明してきました。

本章で作ったプログラムは「レコードメイン」だけを使ったごく簡単なものだったため、上で説明したフローの全体について具体的に理解する必要があまりありませんでした。しかし、複雑なプログラムを作っていくと、思っていた通りに動かない場面に遭遇することもあると思います。そのような時には、上述した Magic エンジンの実行ルールにてらし合わせて考察することにより、理解できるようになると思います。Magic の備えている機能を最大限に生かしたプログラム作りをする上で、Magic エンジンの実行ルールを良く理解しておくことは重要ですので、以後の章の実習を通して、理解を深めていってください。

6. ラインモードオンラインプログラム

前章では、一画面に一レコードづつ表示する、スクリーンモードのオンラインプログラムの作り方について説明しました。

次に、一画面に複数のレコードを表形式で表示するラインモードのオンラインプログラムの作り方とその動作について説明します。例としては、顧客マスターの照会と更新を行うプログラムを使います。下図はこれから作ろうとするプログラムの完成形です。

図 6-1 ラインモード オンラインプログラムの最終形

6.1. APG を使わずに作成する

ラインモードのプログラムもスクリーンモードの場合と同様、APG により簡単に作成することができますが、ここでは勉強のためにゼロから作ってみます。

6.1.1. タスク特性の設定

ラインモードのプログラムでも、実際のところ、ほとんどはスクリーンモードのプログラムと同じです。まずは、新規プログラムを作成し、タスク特性を設定するところから始めます。

1. プログラムリポジトリを開き、F4 で新タスクを作成します。
2. F5 でズームするとタスク特性が開きます。
3. タスク名は「顧客マスター更新」、タスクタイプは「O=オンライン」、初期モードは「M=修正」、メインテーブルは 2 (顧客マスタ) とします。
4. OK を押して、タスク特性を閉じます。

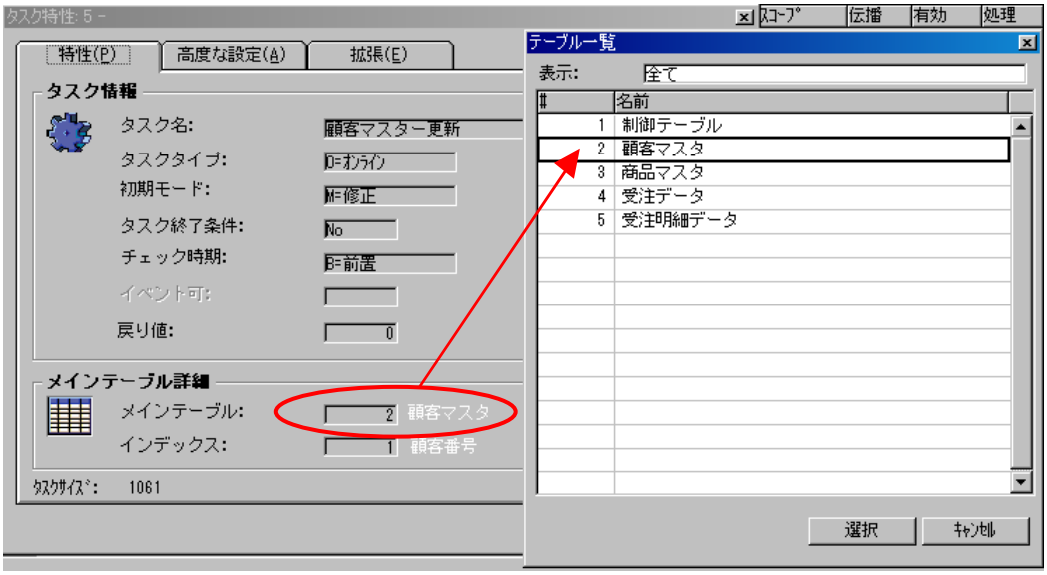


図 6-2 タスク特性の設定

6.1.2. レコードメインの定義

次に、このタスクで利用するデータ項目を定義するため、レコードメインでビューを定義します。

1. レコードメインを開く。
2. セレクトコマンドを使って、全カラムを選択します。

セレクトコマンドの定義方法は、制御テーブル更新の場合と同じなので、ここでは省略します。

図 6-3 レコードメインですべてのカラムを選択

6.1.3. フォームの定義

これまでは、スクリーンモードで作る場合と全く同じでした。ラインモードで異なるのは、フォームの定義です。

1. メニューで「タスク環境(K)」→「フォーム(F)」を選び、フォームテーブルを開きます。(Ctrl+Fでも同じです。)

図 6-4 メニューからフォームテーブルを開く

2. フォームテーブルからF5でズームすると、フォームエディッタが開きます。この時点ではまだフォームは空です。
3. コントロールパレットから「テーブル」をクリックします。マウスの形状が変わります。
4. テーブルを配置したいところへ、そのままマウスを移動します。
5. フォーム上でクリックすると、空のテーブルコントロールが配置されます。

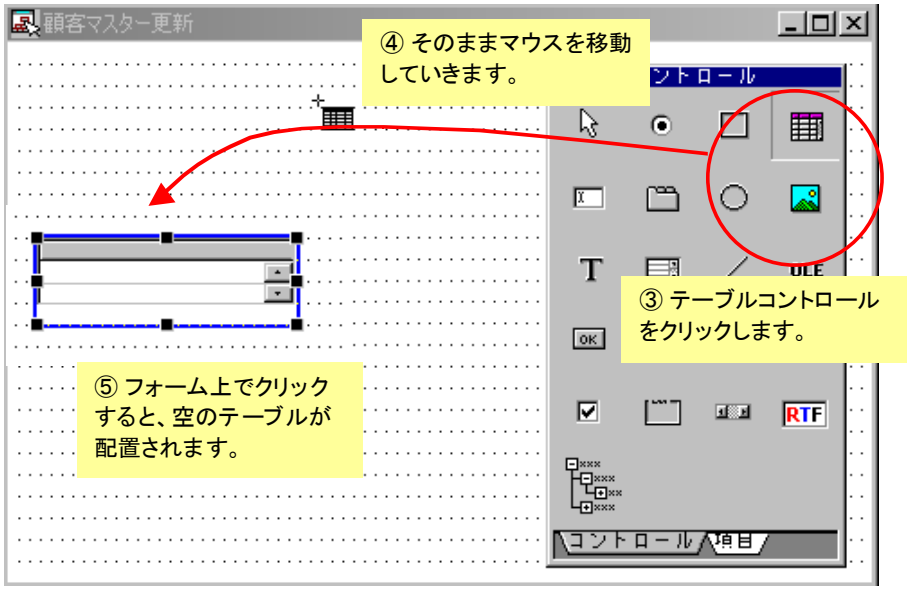


図 6-5 テーブルコントロールを配置

次にテーブル上に項目を貼り付けていきます。

1. 「コントロール」パレットで、「項目」タブをクリックします。項目一覧が表示されます。
2. 項目 A(顧客番号)をクリックします。するとマウスの形状が変わります。
3. そのまま、テーブルコントロールの上でクリックします。テーブルコントロールに、項目 A が配置されます。
4. テーブルの高さが低いので、マウスでテーブルコントロールの底辺をドラッグして、テーブルの高さを調整します。

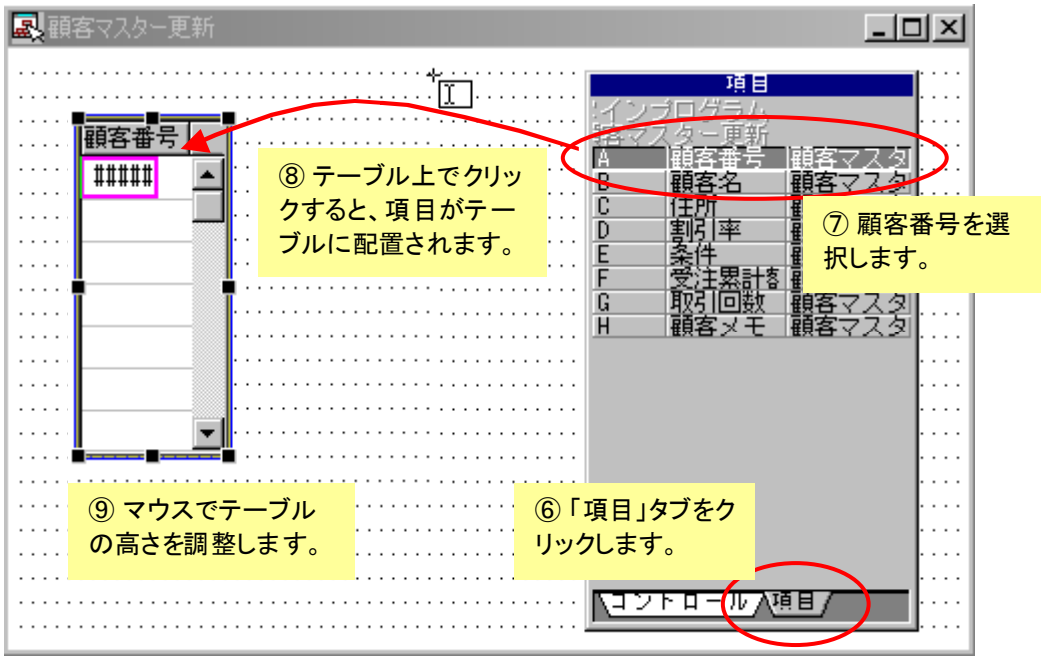


図 6-6 テーブルコントロールにデータ項目を配置 (1)

同様に項目 B をクリックして、テーブルコントロール上で再度クリックします。

項目 A の横に、項目 B が追加されます。テーブルの幅は、新しい項目がきちんと入るように自動的に拡大されます。

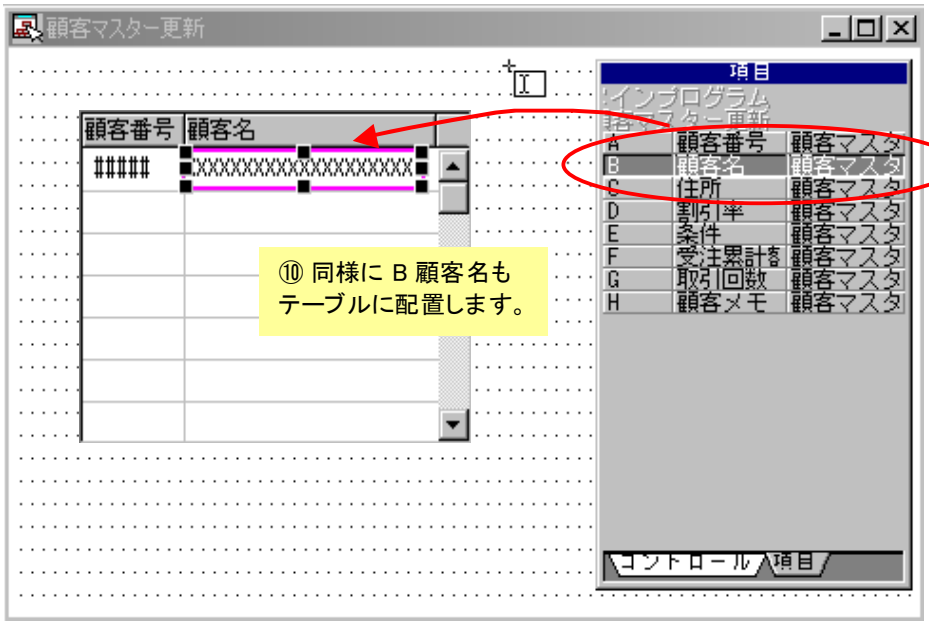


図 6-7 テーブルコントロールにデータ項目を配置（2）

1. 以下同様に、全項目をテーブルコントロールに貼り付けてください。
2. ただし、最後の「顧客メモ」の項目は非常に長いので、テーブルの中に入れようとすると、テーブル全体が非常に横に長くなってしまい、普通のディスプレイの大きさでは収まらなくなってしまいます。このため、顧客メモの項目だけはテーブルの外に表示するようにさせましょう。
   まず、項目パレットで H （顧客メモ） をクリックし、マウスの形状が変わったら、そのまま、今度はテーブルの上ではなく、その下の余白の部分をクリックします。
   そうすれば、この項目がテーブルの外、フォーム上に直接配置されるようになります。

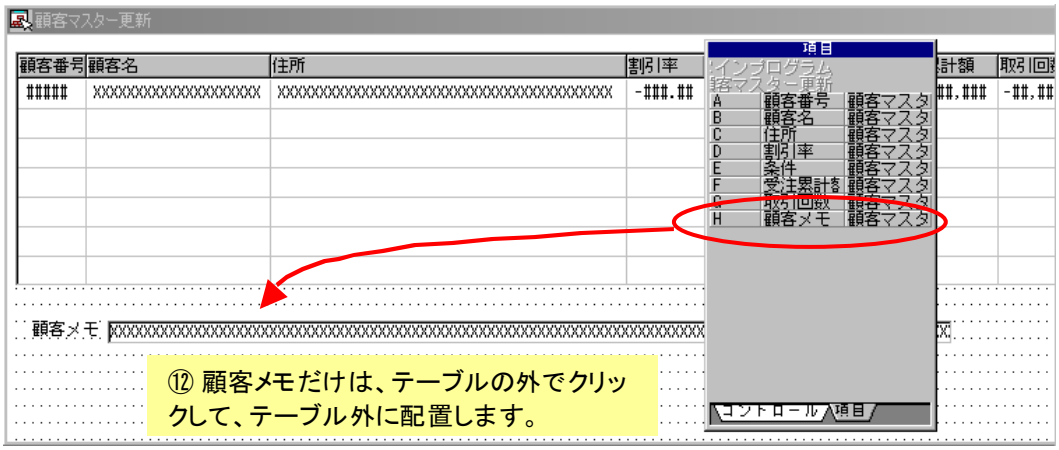


図 6-8 顧客メモをテーブル外に配置

1. 最後に、ウィンドウの幅や、テーブルコントロールの高さなどをマウスで調整してください。

これで、ライン形式のフォームができあがりです。

ESC でフォームエディタを閉じ、Enter 2 回でタスクを閉じて、プログラムリポジトリまで戻ってくだ

さい。

補足： 各項目に対応するエディットコントロールは、テーブルの最初の行にだけしか表示されませんが、実行時にはこれがすべての行にコピーされて、全行についてデータが表示されるようになります。

6.1.4. 実行してみる

1. プログラムリポジトリから、F8でエラーがないことを確認して、F7で実行してみてください。テーブル形式でレコードが表示されます。
2. スクリーンモードの時と同様、データの修正や、F2キーによる修正の取り消しなどを行ってみてください。
3. ↓、↑ キーを使ってレコードの移動することができます。テーブルの最後まで移動して↓キーを押すと、自動的にスクロールします。
   このとき、行を移動すると、「顧客メモ」の表示が変わることに注意してください。
4. スクロールバーをクリック・ドラッグして、スクロールさせることもできます。
5. さらに、PgUp、PgDnキーで、画面単位でのスクロールもできます。
6. Ctrl+Homeや、Ctrl+Endキーにより、先頭レコード、最後のレコードにジャンプすることもできます。

図 6-9 ラインモードでの実行画面（顧客マスター）

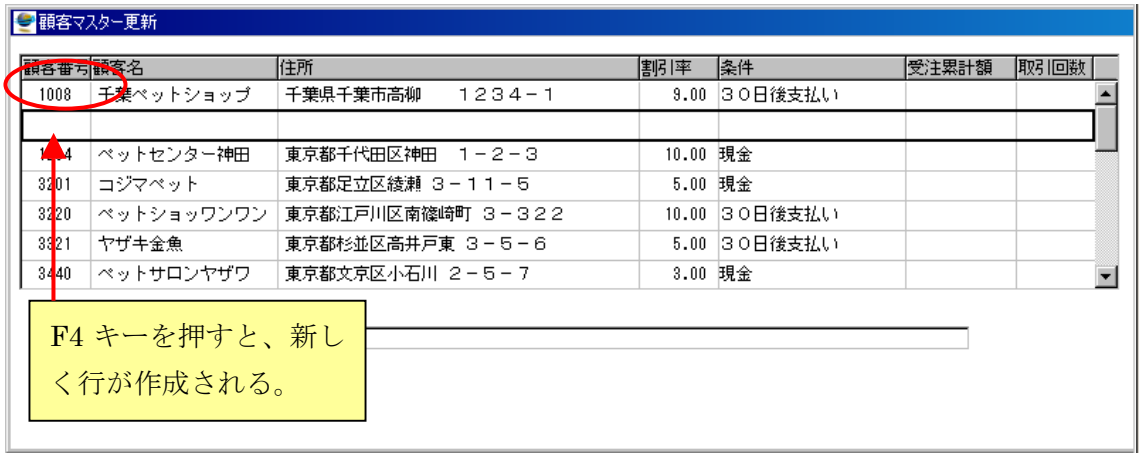
6.2. 実行時のオプション

ここで、Magic の実行エンジンが備えているいろいろな便利な機能について見てみましょう。ここで説明する機能は Magic エンジンが備えているもので、作ったプログラムには一切、手を加えなくても利用することができます。

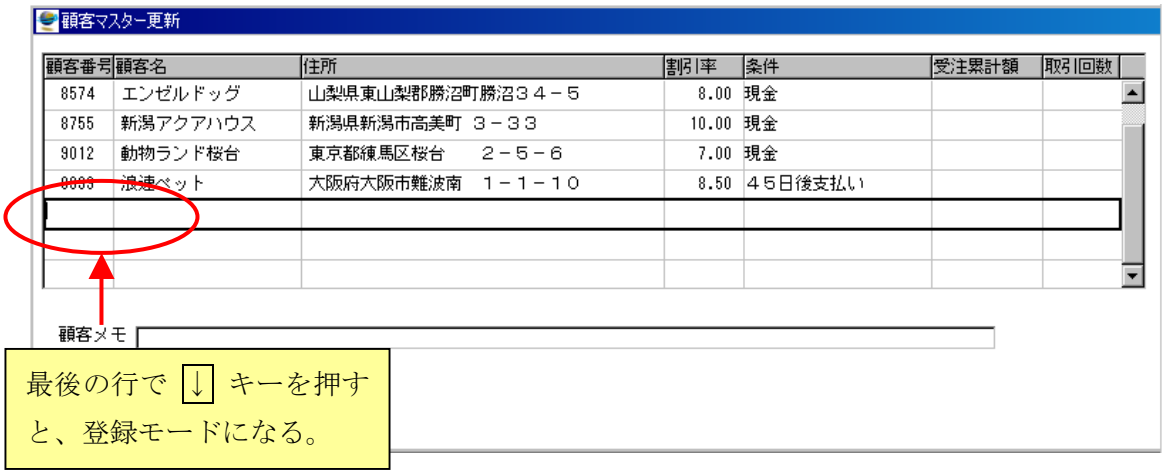
6.2.1. 修正モードでのデータ登録

修正モードでは、新規データの登録も行うことができます。新規データ登録を行うには、次のいずれかの方法があります。

F4 キーで追加： 実行時に、F4 キーを押してください。今カーソルのある行のすぐ下に、新しく空白行が作成されます。ここに適当にデータを入力すれば、新しいレコードが作成され、テーブルに格納されます。



テーブル最後部に移り、↓キーを押す： 実行時、Ctrl + End などを押して、最後のレコードに移動してから、さらに↓キーを押すと、登録モードになって、最後の行の下に新しい行が作成され、新規レコードを登録できるようになります。



6.2.2. モードの変更

顧客マスター更新プログラムの初期モードは修正モードですが、実行時に他のモードに変更することもできます。

照会モード： 照会モードは、データを参照するだけで修正を行わないモードです。読み込み専用のテーブル

を参照する場合に利用します。
照会モードへ変わるには、メニュー「オプション(O)」から「照会(Q)」を選びます。あるいは、Ctrl+Q でも同じです。

図 6-10 メニューから照会モードを選択

登録モード： 登録モードは、新規レコード入力専用のモードです。このため、テーブルにレコードがあったとしても、初期状態ではレコードは 1 件も表示されません。

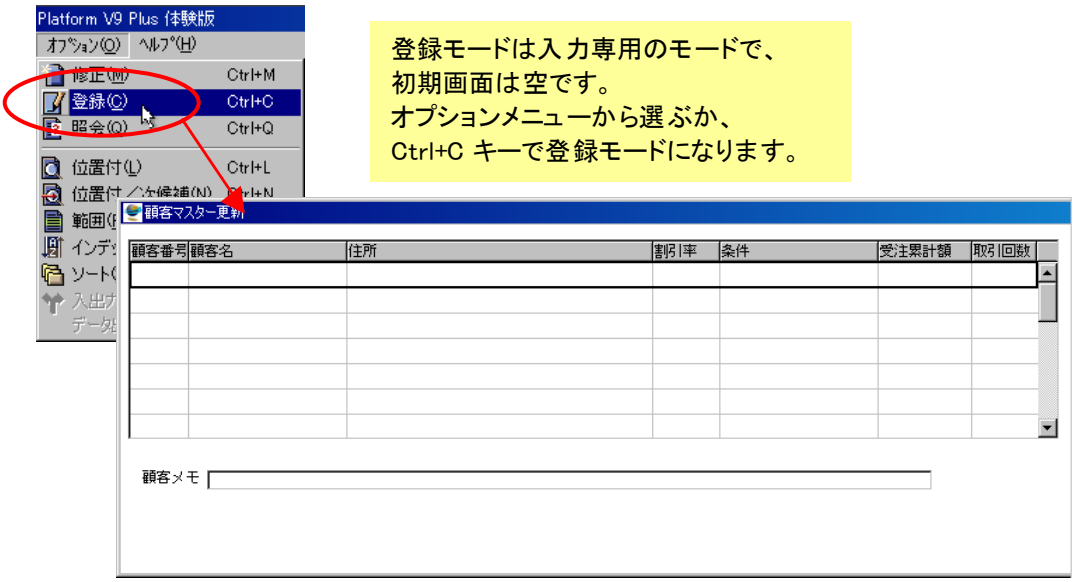


図 6-11 登録モード

登録モードに変わるには、照会モードと同様、メニュー「オプション（O)」から「登録(C)」を選ぶか、あるいは Ctrl + C キーを押します。

修正モード： 修正モードに戻るには、同じく、メニュー「オプション（O)」から「修正(M)」を選ぶか、あるいは Ctrl + M キーを押します。

6.2.3. 範囲指定

顧客マスター更新プログラムでは、顧客マスターテーブルにあるレコードがすべて表示されます。しかし、利用者が実行時に表示するレコードの範囲を絞り込みたいことがあります。このような場合には、実行モードでの範囲指定機能を利用することができます。
例として、顧客番号が 3000 から 5000 までの顧客だけを表示させるようにしてみましょう。

1. まず、顧客マスター更新プログラムを実行してください。
2. 次に、メニュー「オプション(O)」から「範囲(R)」を選びます。すると、テーブルの内容が空白になります。このとき、ステータスバーの表示が「範囲」になっていることを確認してください。
3. 顧客番号に 3000 と入力してください。範囲の最小値になります。
4. メニュー「オプション(O)」から「終了値(T)」を選んでください。あるいは F8 キーでも同じです。このとき、ステータスバーが「終値-範」となっていることを確認してください。
5. 顧客番号に 5000 と入力してください。これが範囲の最大値になります。
6. 範囲の最小値と最大値は、「オプション(O)」メニューの「開始値(F)」と「終了値(T)」、あるいは F7 / F8 キーにより確認できます。
7. Enter キーを押します。すると、指定した範囲に入るレコードのみが表示されます。(次ページの図を参照)
8. 範囲指定をクリアして、全レコードを見たい場合には、範囲指定画面で、メニュー「オプション(O)」から「全項目クリア(C)」あるいは「項目クリア(V)」を選んでください。全項目クリアと項目クリアの違いは、項目クリアではカーソルのあるカラム上に指定されている範囲指定だけがクリアされるのに対し、全項目クリアでは全カラムの範囲指定がすべてクリアされることです。

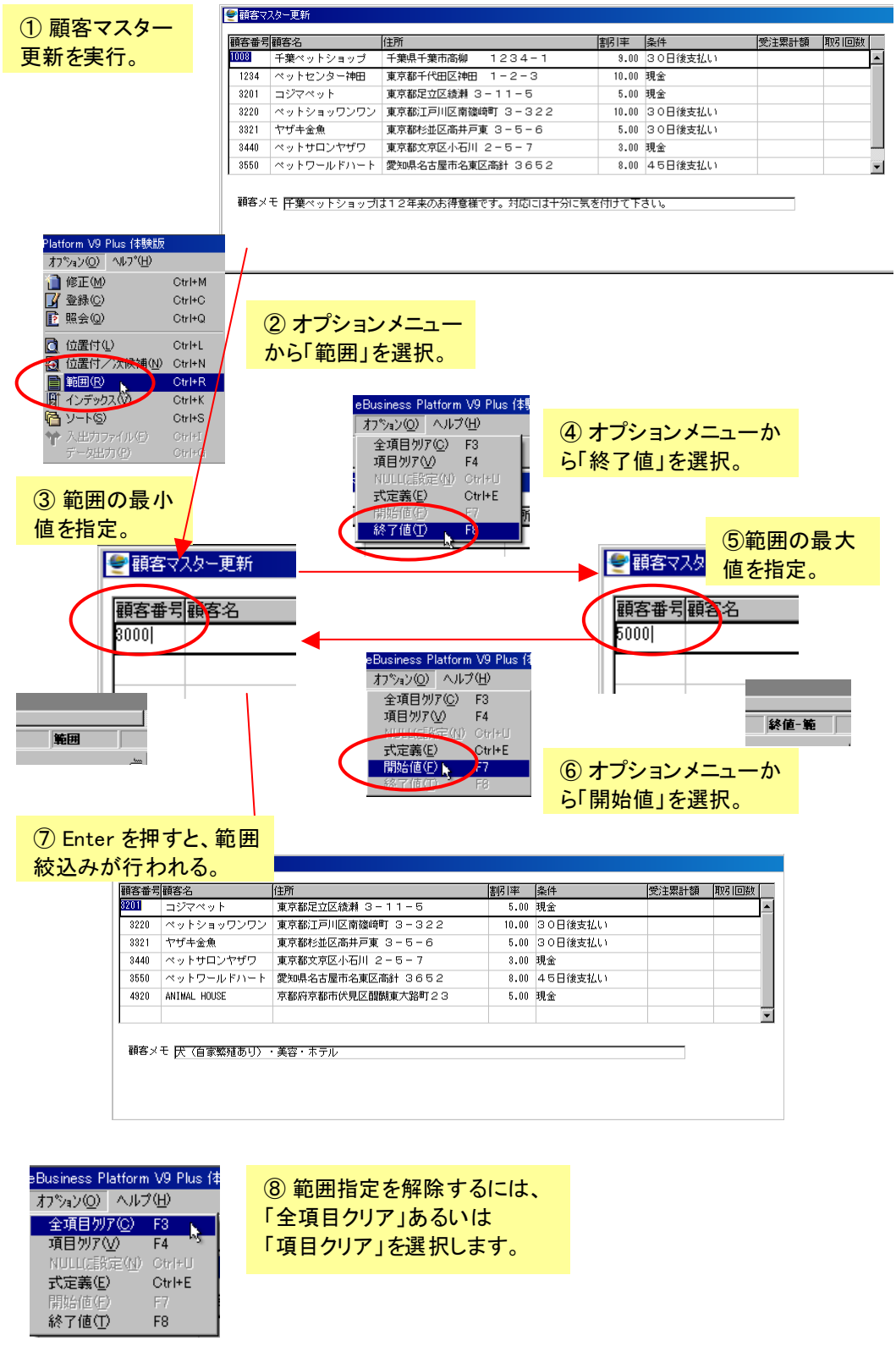


図 6-12 オンラインプログラムでの範囲指定

参考： ここでは顧客番号で範囲指定を行いましたが、名前や住所など、別のカラムで指定することも可能です。また、複数のカラムを同時に指定することも可能で、この場合にはすべての範囲が AND 条件により結合されることになります。

注意： 範囲指定を行う場合、インデックスの定義されているカラムで指定すると、高速に検索することができますが、インデックスの定義されていないカラムで指定すると、全件検索になるので、レコード数が多い場

合には非常に処理が遅くなることがあります。

6.2.4. 式による範囲指定

Magic では、式を使った範囲指定も行うことができます。例として、顧客名の中に「ペット」という文字列がある、という条件で範囲指定することを考えて見ます（図 6-13）。

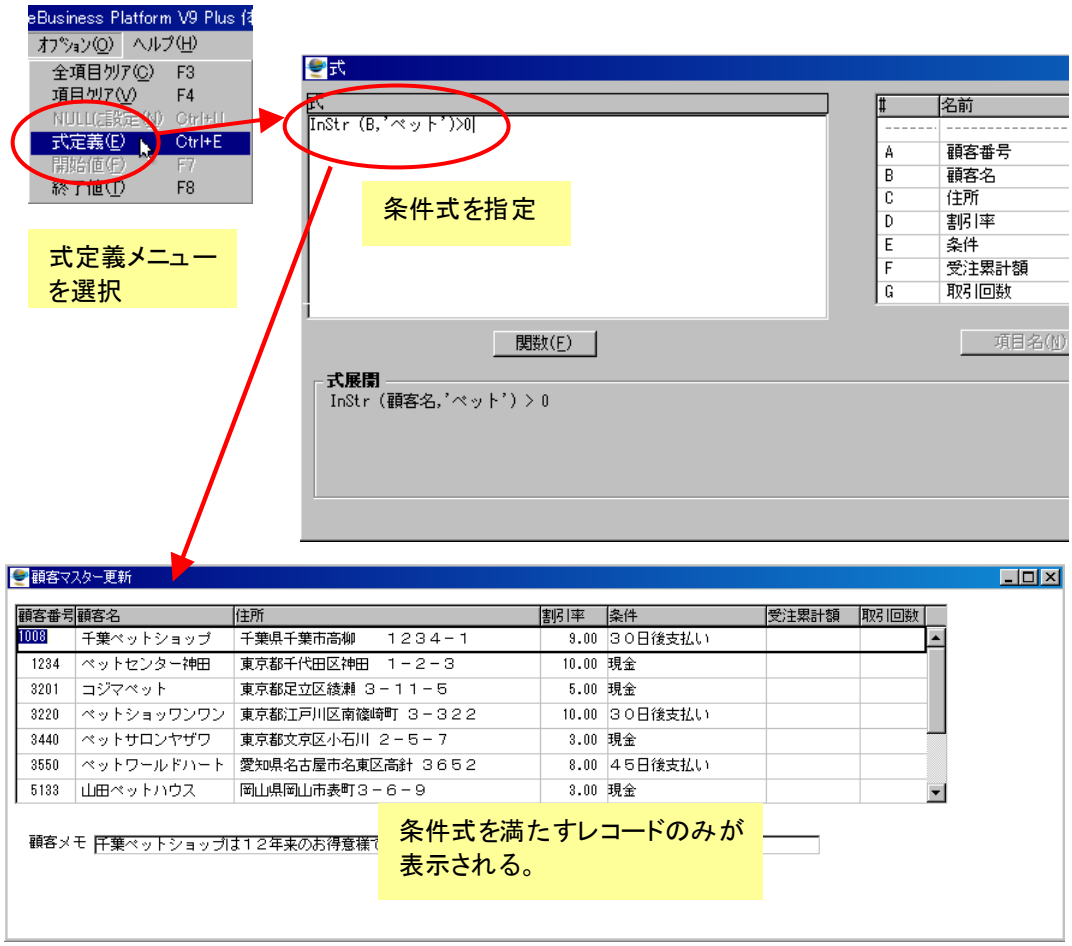


図 6-13 式を使って範囲を指定する

1. 「顧客マスター更新」プログラムを実行します。
2. 先と同じく、メニュー「オプション(O)」から「範囲(R)」を選びます。
3. 範囲の指定画面(空白の画面)で、メニュー「オプション(O)」から「式定義(E)」を選びます。あるいは Ctrl ＋ E でも同じです。
   式定義ダイアログが出るので、Instr (B, 'ペット') > 0 と指定します。
   参考： InStr は、文字列の中の部分文字列を検索する Magic の組み込み関数で、見つかった場合には文字の位置を、見つからなかった場合には 0 を返します。検索条件は論理型で指定する必要があるため、InStr の結果を 0 と比較して、条件としています。また、B というのは、右の一覧にあるように、顧客マスタの顧客名を示す変数記号です。
   入力した後は、人間が見てわかりやすいように、下の「式展開」の欄に、式中の変数記号を実際の名前で置き換えたものが表示されます。
4. OK ボタンをクリックして、範囲の指定画面に戻ります。
5. そのまま Enter キーを押します。顧客一覧が表示されますが、「顧客名」にすべて「ペット」という文字が入っていることを確認してください。

注意： 式で範囲付けをすると、必然的に全件検索となります。レコード数の多いテーブルに対して式の範囲付けを行うと、非常に時間がかかることがあるので、注意してください。

参考： 上の例では、式定義ダイアログで式を直接入力しましたが、関数名やパラメータを忘れてしまった場合のため、関数一覧とヘルプが簡単に出るようになっています（**図 6-14**）

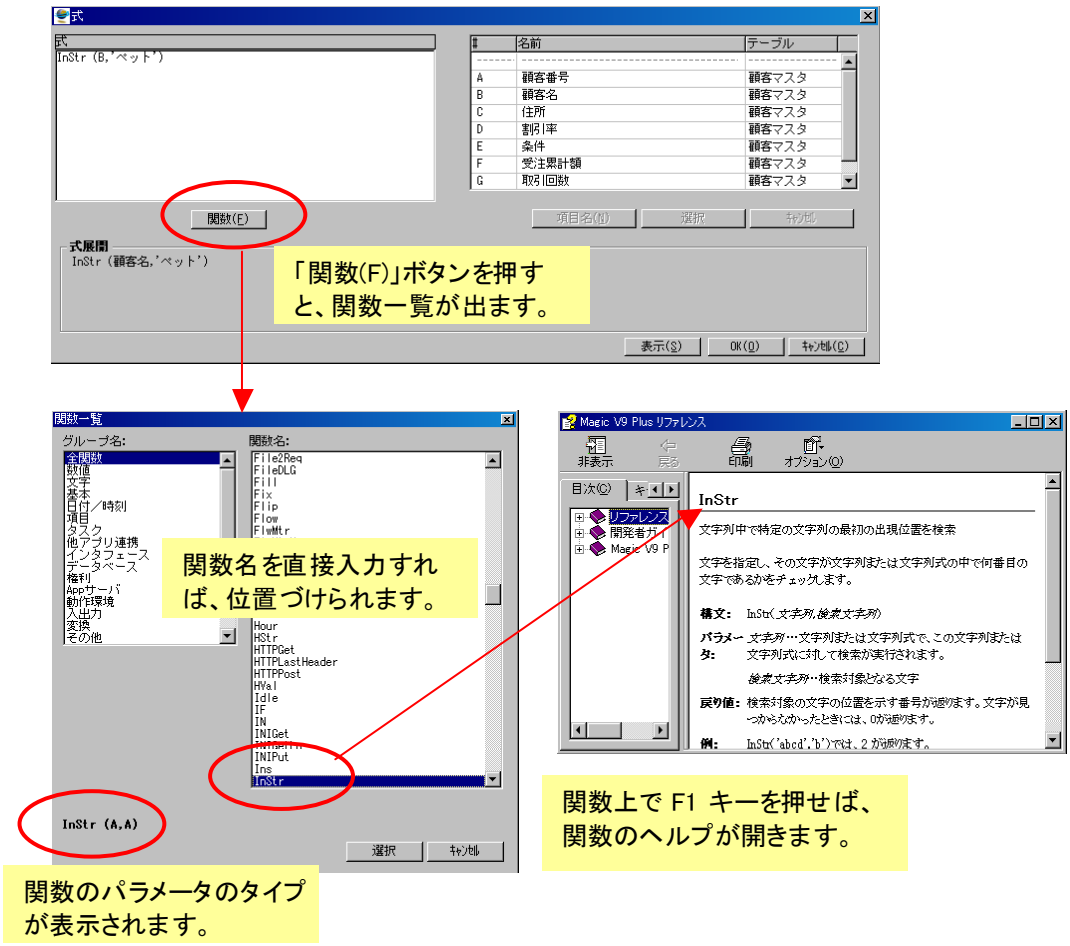


図 6-14 関数一覧とヘルプ

6.2.5. 位置付

範囲指定では、表示対象とするレコードを絞り込みました。しかし、場合によっては、レコード範囲はそのままにして、特定のレコードにジャンプしたい、ということもあります。このようなときには、**「位置付け」**機能を用います。

例として、顧客番号が 5000 のレコードに位置付けを行うときのことを考えて見ます。位置付けの操作方法は、範囲付けの場合と非常に似ています。

1. プログラムリポジトリから、顧客マスター更新プログラムを F7 キーで実行します。
2. メニュー「オプション(O)」から、「位置付(L)」を選びます。(あるいは、Ctlr + L キーでも同じです)。

図 6-15 位置付けメニュー

3. 「顧客番号」カラムに 5000 と入力します。これが位置付けの条件の最小値となります。
4. 範囲指定のときには、続いて最大値も指定しましたが、デフォルトで、最大値も同じ値が設定されています。今回は最小値と最大値が同じでよいので、最大値の指定は行いません。
5. Enter キーを押します。カーソルは顧客番号 5133 の「山田ペットハウス」に位置付けられて、それ以降のレコードが表示されます。

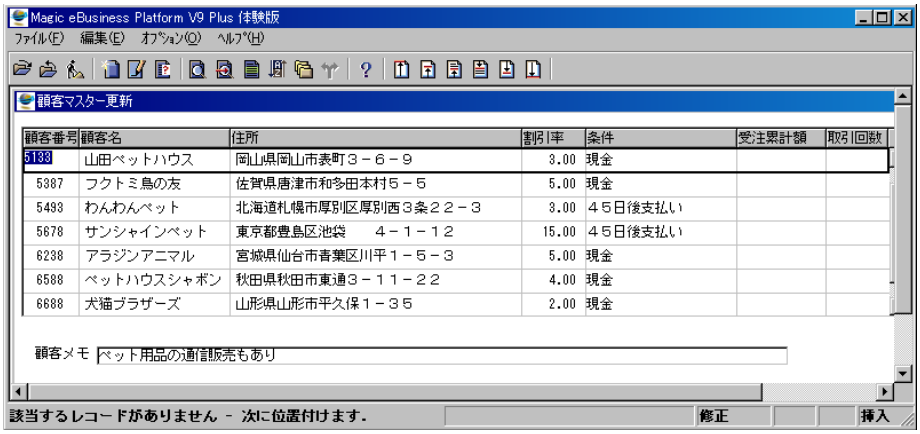


図 6-16 位置付の結果

位置付けは、範囲指定と以下の点で異なるので、確認してください。

- 位置付けの条件は、上限下限とも 5000 としましたが、実際にはこの条件を満たすレコードは存在しません。そのため、ステータスバーに「該当するレコードがありません – 次に位置付けます。」というメッセージが出ますが、画面にはその次のレコード(5133 のレコード)以降が表示されます。範囲指定の場合、該当するレコードが存在しなければ、レコードは 1 件も表示されません。
- 位置付けの後、上下にスクロールして、その周辺のレコードを見ることができます。範囲付けの場合には、指定した範囲の外のレコードは表示されません。

参考： 位置付けの場合でも、範囲指定の場合と同様、式による指定も可能です。

6.2.6. 照会モード位置付

照会モードではデータの修正をできませんが、キーボードからデータをタイプすると、入力した文字にマッチするデータを自動的に検索します。このときの検索の方法は、キー入力するたびに、それまでに入力したものとマッチするレコードを探すという、いわゆるインクレメンタルサーチで検索します。このような検索方法は、テキストエディッタなどではよく見かけますが、Magic ではデータベースに対してこれができます。

また、Magic の照会モード位置付は、文字型の項目では全角文字も使用することができ、また、数値型の場合には、数値としての大小比較を行います。

例として、顧客マスターで住所で検索するときを見てみます（図 6-17）。

1. 顧客マスター更新プログラムを起動します。最初は修正モードで始まります。
2. 照会モードにするため、「オプション(O)」メニューから「照会(Q)」を選びます。
3. カーソルを住所のカラムに移動してください。
4. 仮名漢字変換をオンにして、「東京都」と入力してください。「東京都」にマッチする最初のレコード(顧客番号 1234、ペットセンター神田)にレコードが移動します。
5. 続けて、「練馬区」と入力してください。「東京都練馬区」にマッチするレコード(9012 動物ランド桜台)にレコードが移動します。
6. BackSpace キーを 3 回押してください。「練馬区」の部分が消去されるので、「東京都」にマッチする最初のレコード(ペットセンター神田)に戻ります。

注意： 照会モード位置付では、文字が 1 文字入力されるたびにレコード検索が起こります。カラムにインデックスが定義されていない場合には、検索に非常に時間がかかるので注意してください。

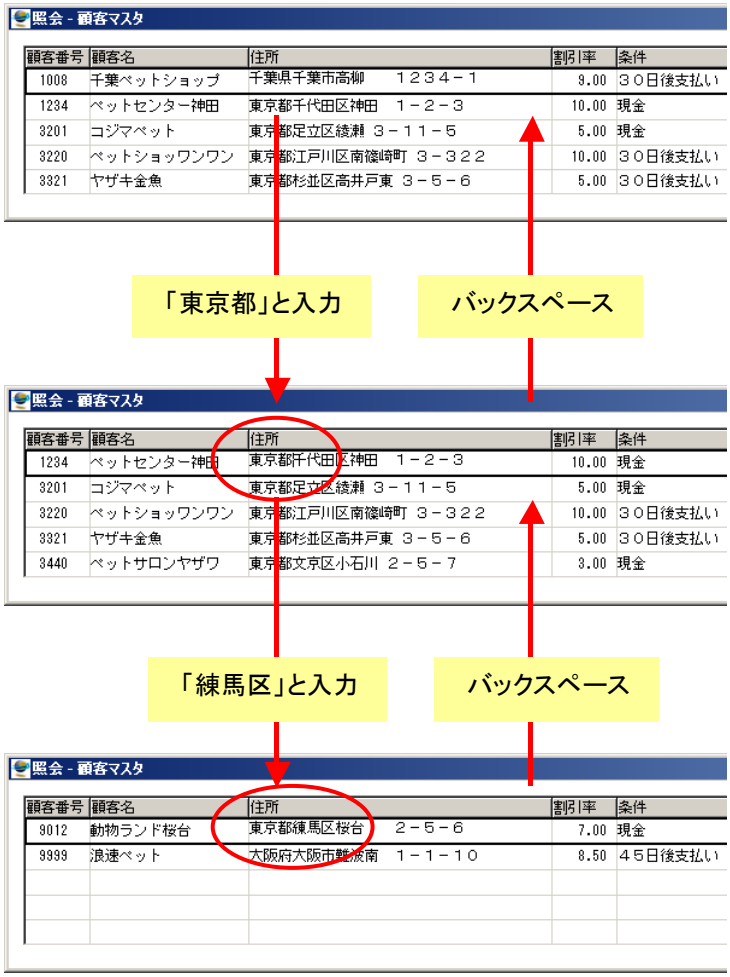


図 6-17 照会モードでの照会モード位置付

6.2.7. 登録モードでのキー重複チェック

登録モードで新規にレコードを作成するとき、間違ってすでに存在するキー値を入力してしまうことがありま

す。このような場合、Magic はキー値を随時チェックして、ユーザが全カラムのデータを入力する前に、エラーメッセージを出して修正を求めます。
これも顧客マスター更新のプログラムで確認してみましょう。

1. 顧客マスター更新プログラムを起動します。最初は修正モードで始まります。
2. F4 で新規行を作成してください。空白の行が作成され、一時的に登録モードになります。
3. 「顧客番号」に 1234 と入力してください。(1234 という顧客はすでにテーブルに存在してます。)
4. Tab キーなどで、次のカラムに行こうとすると、ステータス行に「インデックスが重複しています。テーブル： 顧客.MST」と出ます。

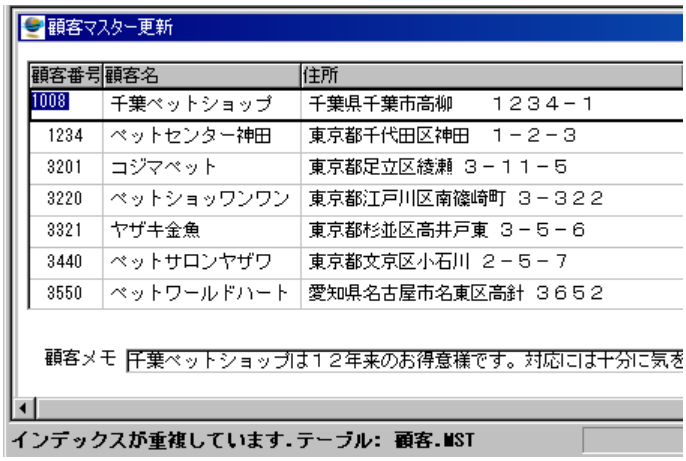


図 6-18 キー重複時のエラーメッセージ

このように、キーの重複チェックはできるだけ早い時期に自動的に行われるので、ユーザが無駄な入力をしないですむようになります。

6.2.8. 表示順の変更

Magic のプログラムでのレコード表示の順番は、タスク特性の「インデックス」で指定されたインデックスの順番になります。顧客マスター更新プログラムでは、「顧客番号」インデックスが指定されていたので、顧客番号の順にレコードが表示されます。
レコードの表示順を実行時に変えることもできます。これには、別のインデックスを指定する方法と、ソートを行う方法とがあります。

① インデックスを指定する方法

インデックスが複数定義されている場合に、タスク特性で指定したインデックス以外のインデックスを選択することができます。

1. 顧客マスター更新プログラムを起動します。最初は顧客番号の順にレコードが表示されます。ここでメニュー「オプション(O)」から、「インデックス(V)」を選びます(あるいは、Ctrl + K キーを押します)。
2. インデックス一覧が表示されるので、2 番目のインデックス(顧客名)を選び、選択 ボタンを押します。
3. 顧客名の順(漢字コードの順になります)でレコードが表示されます。

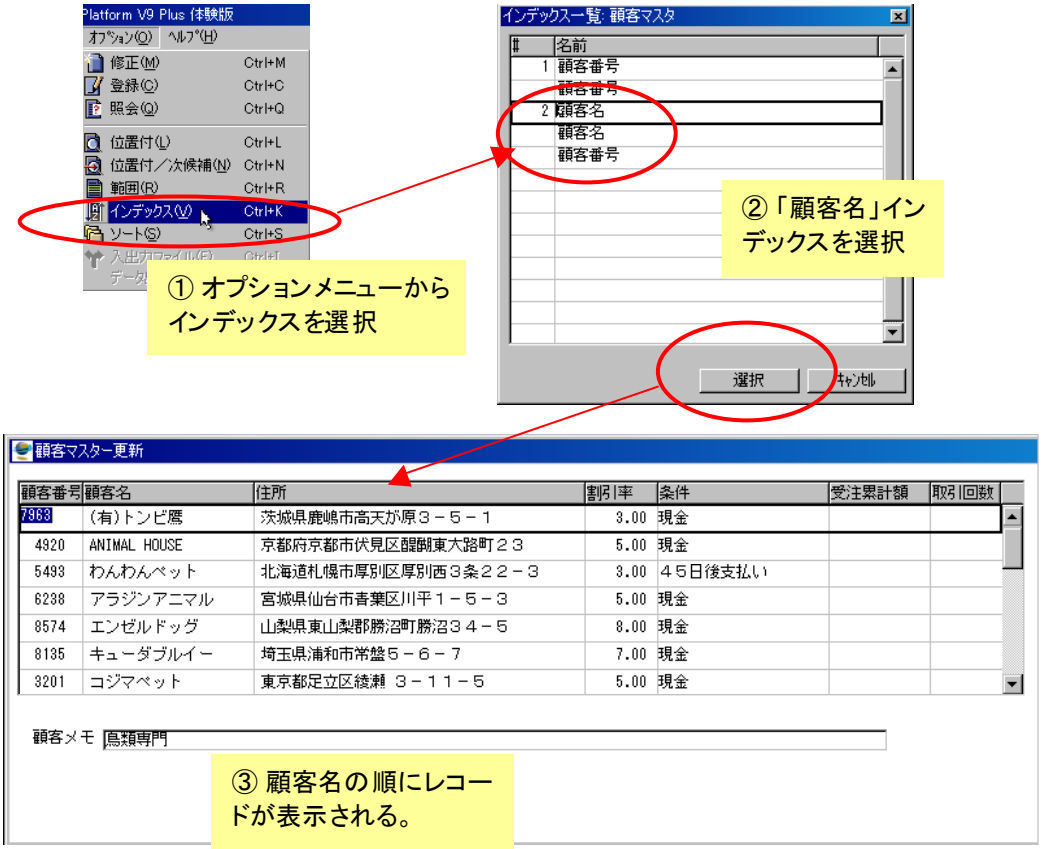


図 6-19 インデックス指定により表示順を変更

② ソートを利用する方法

インデックスが定義されていないカラムを使って順序付けをしたい場合には、ソートを使います。割引率の大きい順に顧客を表示することを考えて見ましょう。

1. 顧客マスター更新プログラムを起動します。最初は顧客番号の順にレコードが表示されます。ここでメニュー「オプション(O)」から、「ソート(S)」を選びます(あるいは、Ctrl + S キーを押します)。
2. 「項目」に D (割引率)を指定します。また、割引率の大きい順に表示させたいので、「順」は「D=降順」とします。
3. OK ボタンを押すと、割引率の順にレコードが表示されます。

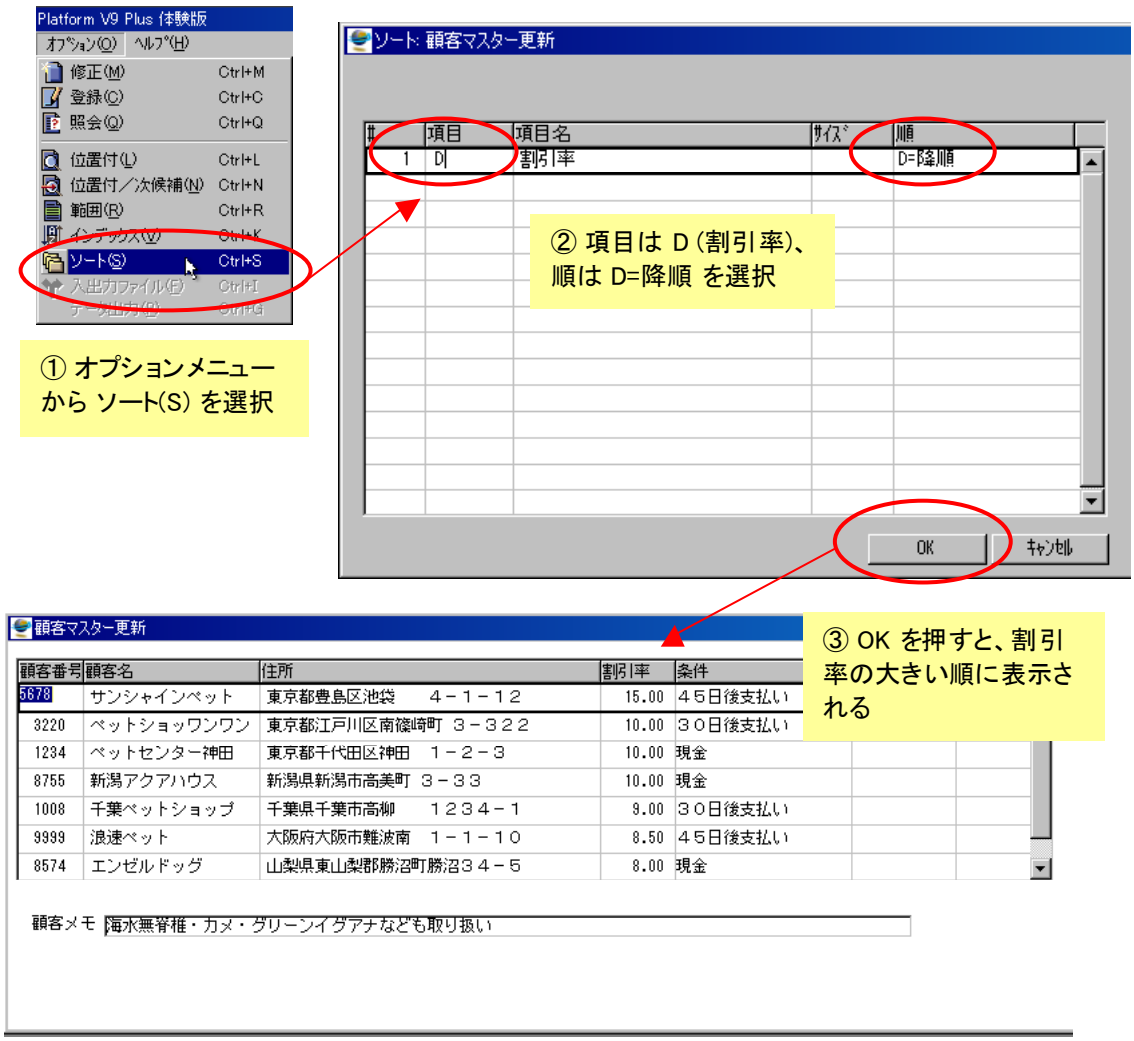


図 6-20 ソート指定による表示順の変更

6.3. 本章のまとめ

本章では、ラインモードのオンラインプログラムを通して、次のことを学びました。

- ラインモードのフォームの作成方法。
- ラインモードのオンラインプログラムは、スクリーンモードと比べ、フォームが異なるだけで、レコードメインなどその他の部分は全く同じであること。
- Magic の実行エンジンが備えているさまざまなオプション機能とその使い方。

本章の内容を応用して、商品テーブル更新タスクも作成してください。

7. ボタンとイベント

使いやすいユーザインターフェースを作るには、エディットコントロールのほかに、ボタンコントロールをうまく活用することが必要です。また、Windows のアプリケーションはイベントドリブンの形でプログラムを作るのが一番自然なので、ボタンとイベントとをうまく活用してプログラムを作ることが、良いユーザインターフェースを作成する基本になります。

本章では、Magic におけるボタンとイベントハンドリングの基本について見ていきます。例として、制御テーブル更新に「終了」ボタンをつけることにします。下図は、制御テーブル更新プログラムの最終形です。

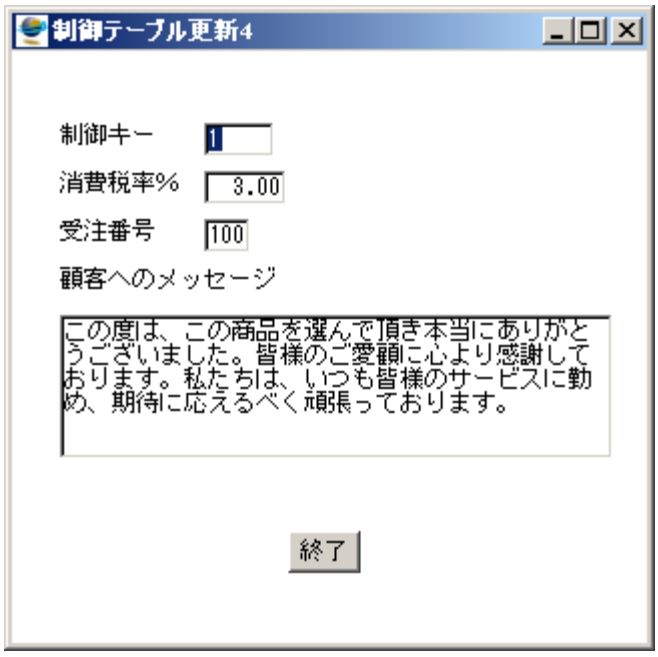


図 7-1 制御テーブル更新の最終形

参考情報： 本章の内容については、リファレンスマニュアルにより詳細な情報がありますので参考にしてください。	
イベント全般	第 5 章 Magic アプリケーションエンジン → イベントハンドリング
プッシュボタン	第 9 章 表示フォーム → GUI 表示コントロール → プッシュボタンコントロール 第 9 章 表示フォーム → GUI 表示コントロール特性 → プッシュボタンコントロール特性

7.1. フォームにボタンを貼り付ける

最初に、制御テーブル更新プログラムにボタンを貼り付けてみましょう。

（1）プログラムのコピー

先に作った制御テーブル更新プログラムをそのまま使っても良いのですが、ここではコピーして使うことにしましょう。

1. プログラムリポジトリを開き、カーソルは最後の行に置いた状態で、「編集(E)」メニューより、「複写登録(Y)」を選びます。 Ctrl +R キーを押しても同じです。
2. 複写登録ダイアログが開くので、コピー元のプログラムの番号(4 番)を、「開始番号」「終了番号」ともに指定します。これにより、プログラムの 4 番がコピーされます。
3. OK ボタンを押すと、最下行の次に「制御テーブル更新」プログラムがコピーされます。

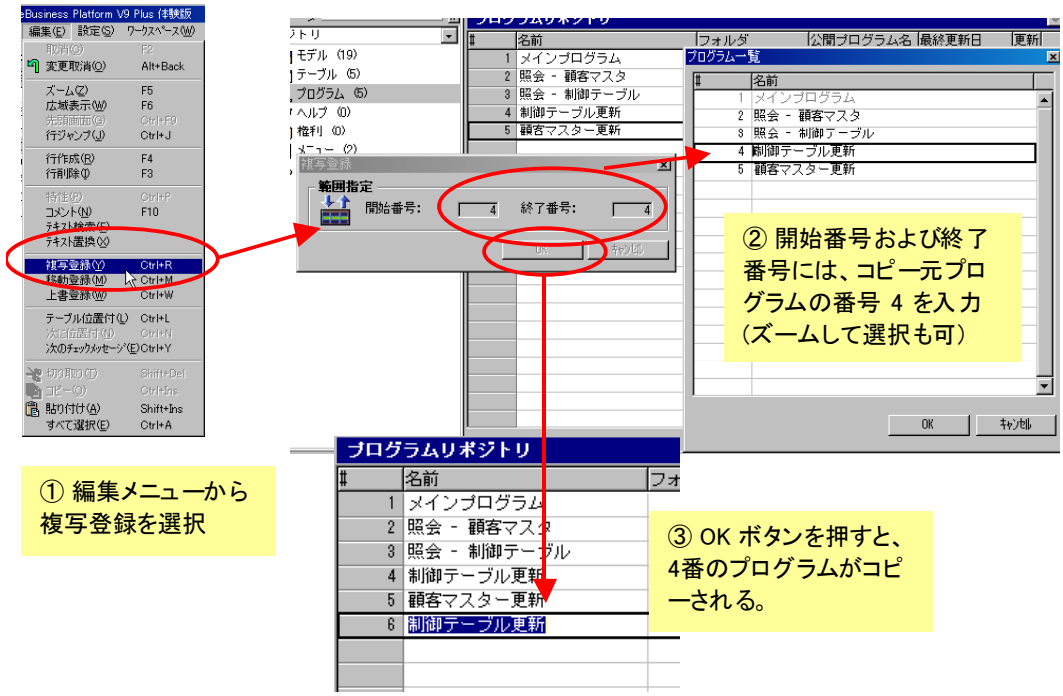


図 7-2 プログラムのコピー

（2）ボタンの貼り付け

次に、プログラムを開いて、フォームにボタンを貼り付けましょう。

1. F5 キーでプログラムを開きます。
2. フォームエディッタを開きます。(「タスク環境」メニューから「フォーム」を選ぶか、あるいは Ctrl + F を押すとフォームテーブルが開くので、 F5 キーでフォームを開きます。)
3. コントロールパレットからボタンをクリックします。マウスの形状が変わります。
4. フォーム上のボタンを配置したい場所でマウスクリックします。ボタンが配置されます。

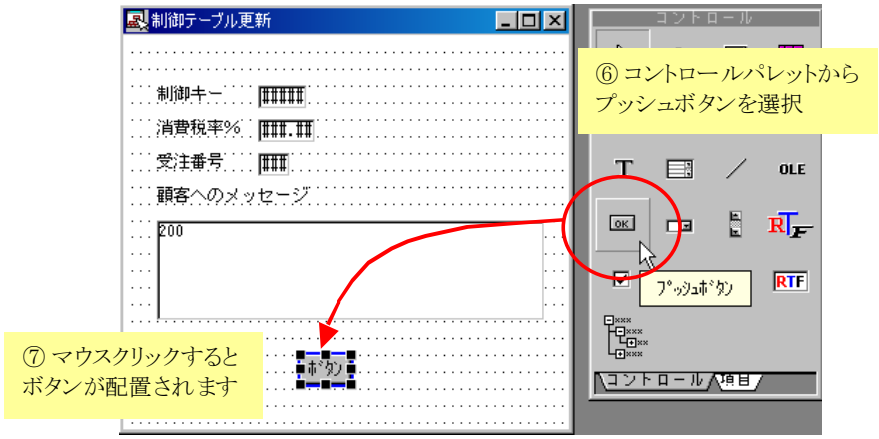


図 7-3 フォーム上へボタンを配置

（3）ボタンのラベルを「終了」にする

1. プッシュボタンをマウスクリックで選びます。プロパティシートにプッシュボタンのプロパティが表示されます。
2. 「書式」プロパティを「終了」とします。

（4）クリック時に発生するイベントの指定

3. プロパティシート上の「実行イベント」プロパティで、ズームします。 F5 キーを押します)。「イベント」ダイアログが表示されます。
4. 「イベントタイプ」は「I=内部」にします。
5. 「イベント」でズームします。アクション一覧ダイアログが表示されます。
6. 「クローズ(C)」を選び、 選択 ボタンを押す（または Enter キーを押す）と、「イベント」ダイアログに戻り、「イベント」には「C:クローズ」が設定されます。
7. OK をクリックする(または Enter キーを押す)と、プロパティシートに戻り、「実行イベント」に「クローズ(C)」が設定されます。

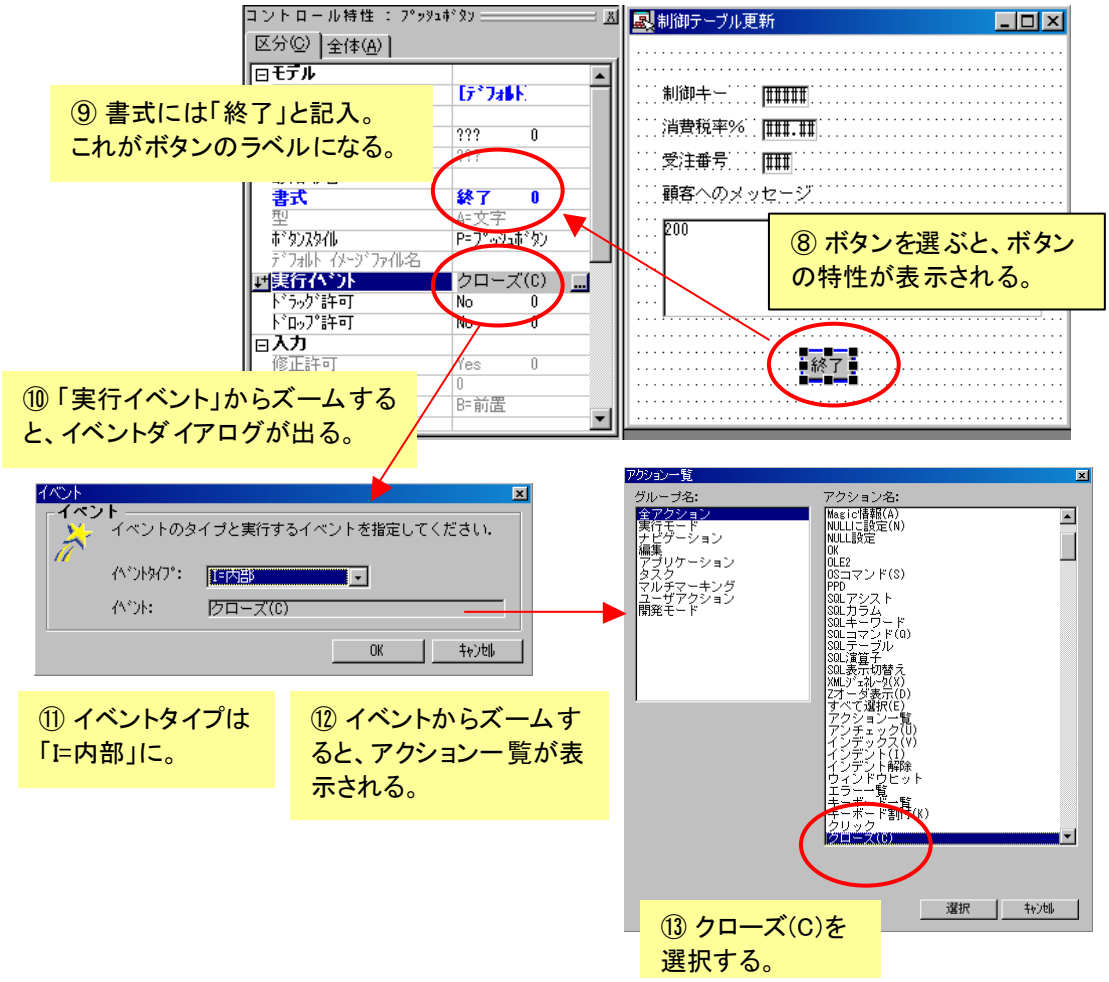


図 7-4 ボタンクリック時のイベントの指定

これにより、プッシュボタンが押されたら「クローズ(C)」イベントが発生するようになります。クローズイベントというのは、内部イベント（Magic エンジンが処理をするイベント）で、開いているウィンドウを閉じる効果があります。

（5）プログラムを閉じる

8. フォームエディッタ、タスクを閉じて、プログラムリポジトリに戻ってください。

（6）実行する

プログラムリポジトリで F7 キーを押して、実行してください。 終了 ボタンを押すとプログラムが終了してウィンドウが閉じることを確認してください。

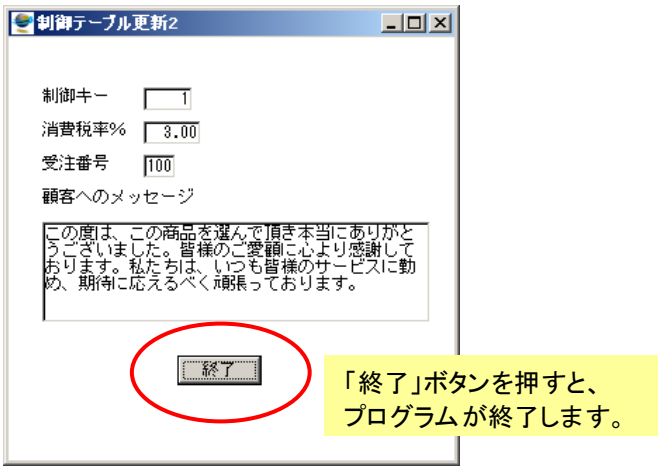


図 7-5 終了ボタンのある画面

ここで、実行時に、Tab を何回か押してください。カーソルが項目間を移動しますが、「終了」ボタンにはパークしません。次には、「終了」ボタンにカーソルがパークするようにしましょう。

7.2. 変数にプッシュボタンを関連づける

Magic では、フォーム上でのカーソルの動きは、レコードメインでの定義と密接に関係しています。一般規則は、次の通りです。

①. レコードメインで定義されているデータ項目が関連付けられているコントロールにのみ、カーソルがパークする。

②. カーソルのパークする順序は、レコードメインでのデータ項目の定義を行うセレクトコマンドの定義順に従う。

③. セレクトコマンドに「条件欄」があるが、この値が No になっているか、論理式で False になる場合には、パークしない。

前節で作成したプログラムでは、ボタンをフォームに貼り付けました。この場合、ボタンがクリックされると、指定されたイベント(クローズ(C))が発生はしますが、対応するデータ項目がレコードメインで定義されていないため、上の規則 ① により、カーソルはパークしません。

ここでは、プッシュボタンに対応するデータ項目をレコードメインに定義することにより、カーソルがパークするように作り直して見ます。

（1）プログラムをコピーします

1. プログラムリポジトリを開き、最下行で「制御テーブル更新」プログラムをコピーし、名前を「制御テーブル更新２」とします。
2. コピーしたプログラムを F5 で開きます。

（2）プッシュボタン用のデータ項目定義

レコードメインに、プッシュボタン用のデータ項目を追加します。このデータ項目は、DBMS 中のレコードと対応しているデータではないので、メモリのみに存在する、変数項目とします。

3. レコードメインを開き、最後の行に移ります。
4. F4 で新規行を作成します。
5. コマンドを「セレクト」とします。(行の先頭で、ズームしてコンボボックスから選択するか、あるいはキーボードから s と入力します。)
6. 「R=実データ」を「V=変数」に変えます。これでこのデータ項目が変数項目として宣言されます。
7. 次の「1」の欄でズームします。「変数項目」テーブルが表示されます。
8. #1 の行で、名前は「終了ボタン」、型は「A=文字」、書式は「8」とします。
9. 変数項目テーブルを開いたまま、プロパティシート「ローカル変数特性」の「GUI 表示」の欄をクリックしてください。初期状態では、「エディット」となっています。これは、フォーム上で表示されるコントロールの形式を表しています。
10. コンボボックスから、「P=プッシュボタン」を選んでください。
11. 右にある … ボタンをクリックしてください。「コントロール特性： プッシュボタン」というプロパティシートが開きます。
12. プッシュボタンの「実行イベント」でズーム（ダブルクリック、あるいは F5)します。「イベント」ダイアログが出てきます。
13. 前節で設定したのと同じ要領で、イベントタイプは「I=内部」、イベントは「クローズ(C)」を選びます。
14. 「ローカル変数特性」のプロパティシートに戻り、「デフォルト値」欄を「終了」にします。

15. Enter キーを 2 回押して、レコードメインに戻ります。

以上の定義により、この項目をフォームに貼り付けたときには、プッシュボタンとして配置され、実行時にこのプッシュボタンがクリックされたときには、「クローズ」イベントが発行されるようになります。

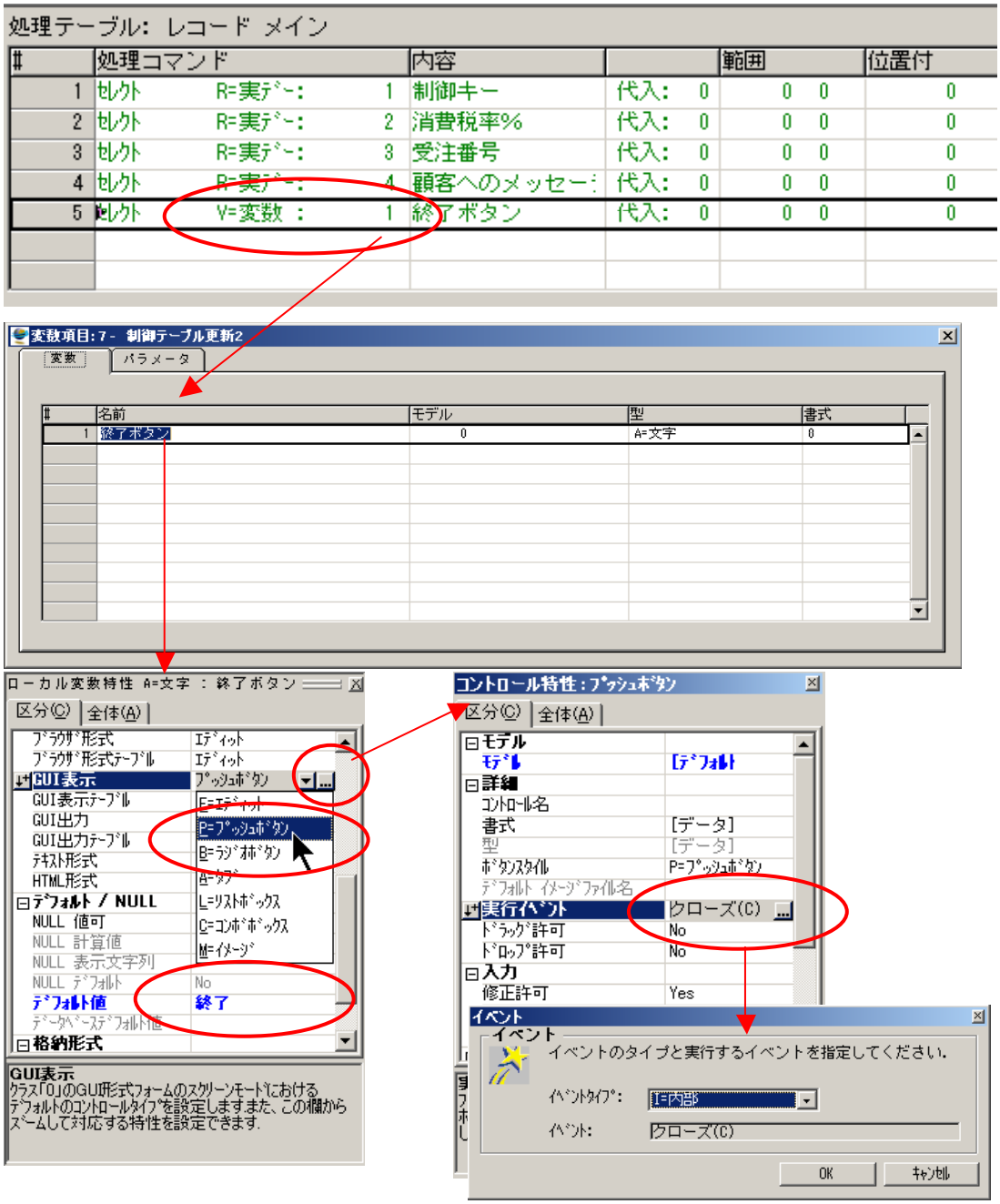


図 7-6 変数にプッシュボタンを割り当て、クリック時にクローズイベントを発行する

（3）フォームへの配置

次には、この変数項目をフォームに貼り付けます。

16. フォームテーブルを開きます（「タスク環境」メニュー →「フォーム」を選ぶか、 Ctrl + F キーで行います）
17. ズームしてフォームエディッタを開きます。

18. 「コントロール」パレットで「項目」タブをクリックし、項目一覧から「終了ボタン」をフォーム上に貼り付けます。プッシュボタンが配置されるはずですので、プロパティシートで、データが E(「終了ボタン」変数項目)、実行イベントが「クローズ(C)」になっていることを確認してください。
19. フォームエディッタを閉じ (ESC キーを押します)、タスクを閉じます。

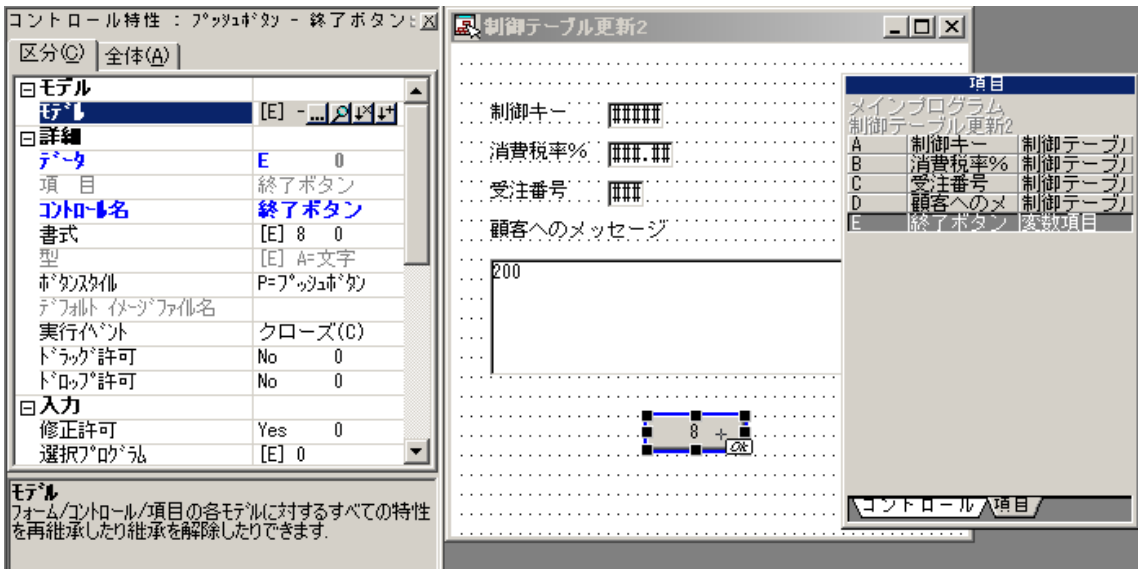


図 7-7 変数をフォームに配置すると、プッシュボタンとして配置される。

（4）実行してみる

プログラムリポジトリから F7 で実行してみて、Tab キーで「終了」ボタンにカーソルがとまること、ボタンを押すとタスクが閉じることを確認してください。

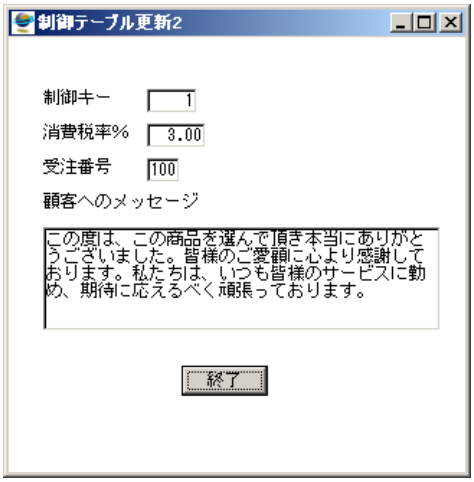


図 7-8 パークするプッシュボタンのある画面

7.3. モデルに登録する。

「終了」ボタンのようにどこでも使うボタンを定義するのに、いちいち前節で書いたような定義をしなければならないのでは不便です。

Magic のモデル機能を用いれば、アプリケーションでよく使うボタン項目の定義を容易にすることができますので、ここではモデルへの登録方法と、プログラムでの利用方法について見ていきます。

7.3.1. プッシュボタンモデルの登録

まず最初に、「終了」プッシュボタンコントロールをモデルリポジトリに定義します。

1. モデルリポジトリを開きます（メニューで「ワークスペース(W)」から「モデル(O)」を選ぶか、あるいは Shift ＋ F3 キーを押します。）
2. 最下行で F4 キーを押し、新規行を作成します。
3. 名前は「終了ボタン」、クラスは「D=GUI 表示」、型は「P=プッシュボタン」とします。
4. プロパティシート「コントロール特性： プッシュボタン」で、書式を「終了」とします。これはボタンのラベルになります。
5. 実行イベント欄でズームします。イベントダイアログが開きます。
6. イベントタイプは「I=内部」、イベントは「クローズ(C)」とします。
7. モデルリポジトリに戻ります。

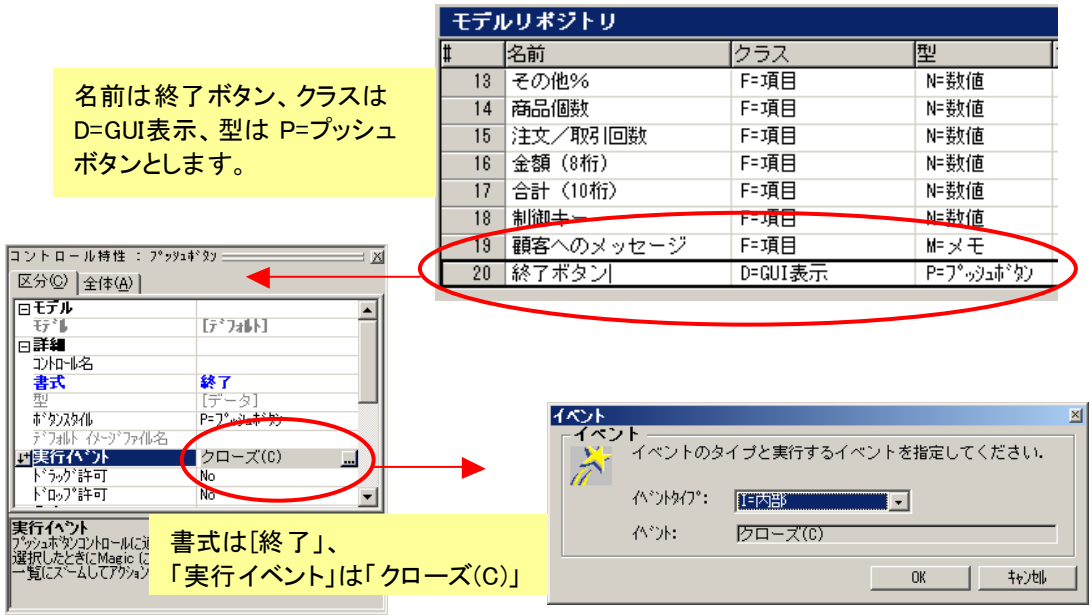


図 7-9 「終了」プッシュボタンのモデルの登録

7.3.2. 項目モデルの登録

次には、このプッシュボタンに関連づけられた、データ項目モデルを定義しましょう。

1. モデルリポジトリで、 新規行を作成します。(最下行で F4 キーを押します)。
2. 名前は「終了ボタン項目」、クラスは「F=項目」、型は「A=文字」とします。
3. プロパティシート「項目特性： A=文字」に移り、書式を「8」、デフォルト値を「終了」とします。
4. GUI 表示特性を「プッシュボタン」にします。

5. そこからズームします。「コントロール特性： プッシュボタン」のプロパティシートが開きます。
6. 「モデル」特性からズームします。モデル一覧が表示されます。
7. 「終了ボタン」を選択します。
8. モデルリポジトリまで戻ります。

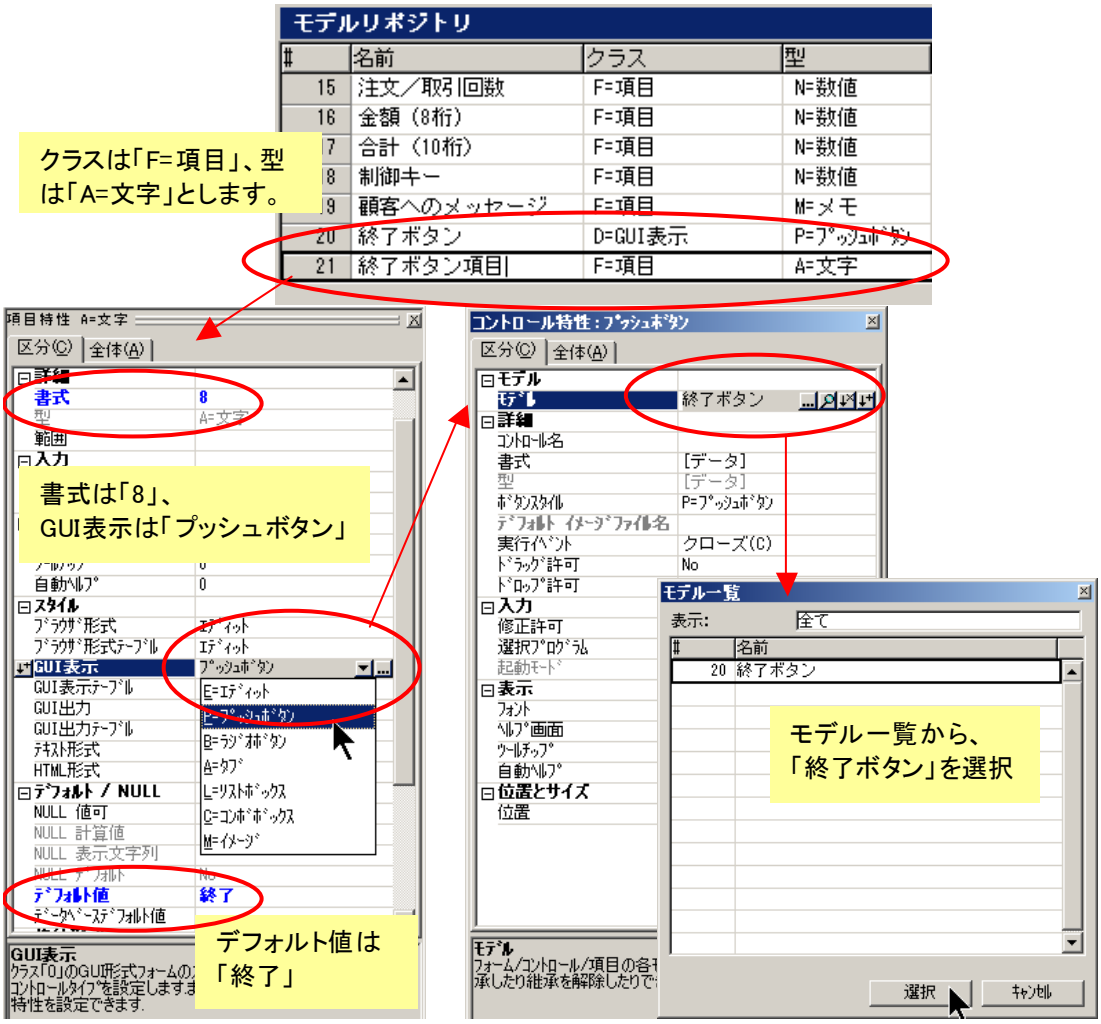


図 7-10 「終了」プッシュボタンに関連づけられた項目モデルの登録

このモデルは、次のような定義となっています。

- 最大８バイトの長さを持つ文字型のデータ項目のモデルである。
- フォームに配置されたときにはプッシュボタンとして配置される。
- 実行時にクリックされたときには、クローズイベントが発行される。

このモデルではプッシュボタンのモデル「終了ボタン」を参照しています。クリック時に発行されるイベントは「クローズ(C)」ですが、これは「終了ボタン」モデルから継承しています。

7.3.3. プッシュボタンモデルをタスクで利用

まずは、「終了ボタン」プッシュボタンモデルをタスクで利用してみましょう。

1. プログラムリポジトリで「制御テーブル更新」プログラムをコピーし、「制御テーブル更新 3」と改名し

ます。

2. プログラムを開き、さらにフォームエディッタを開きます。
3. 前の例のときと同様に、コントロールパレットからプッシュボタンをフォームに貼り付けます。
4. プッシュボタンプロパティの「モデル」でズーム → モデル一覧
5. 「GUI_終了ボタン」を選びます。
6. フォームエディッタ、タスクを閉じます。

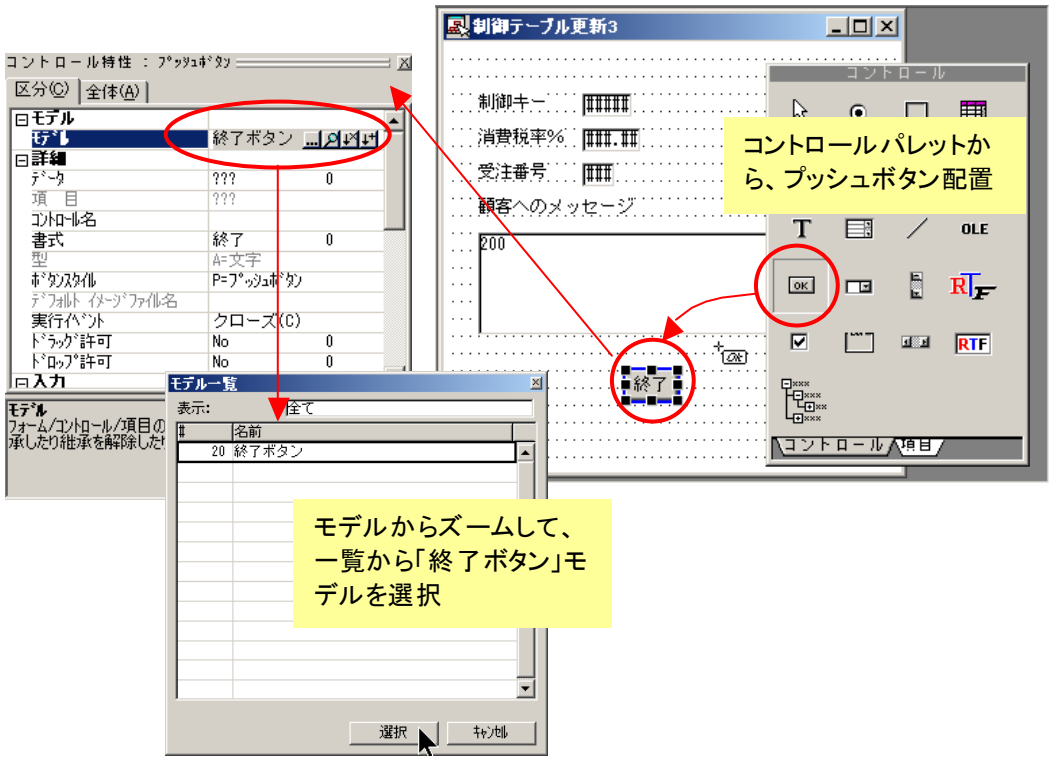


図 7-11 プッシュボタンモデルをプログラムで参照

プログラムを実行して、動作を確認してください。
このプログラムでは、「制御テーブル更新１」の場合と同じく、プッシュボタンに対応するデータ項目がレコードメインに定義されていないので、プッシュボタンにはカーソルはパークしません。

参考： フォームエディッタ上で、終了ボタンモデルを参照するプッシュボタンを配置する場合、もっと簡単にすることもできます。

1. フォームエディッタを開きます。
2. コントロールパレット上のプッシュボタンのところを、マウスの右ボタンでクリックします。すると、モデルリポジトリで定義されているプッシュボタンモデルの一覧が表示されます。この例では、「終了ボタン」というプッシュボタンモデルが定義されているので、[default] に並んで表示されます。
3. 「終了ボタン」をクリックすると、マウスのカーソル形状が変わります。
4. そのままフォーム上でクリックします。「終了ボタン」をモデルとするプッシュボタンコントロールが配置されます。

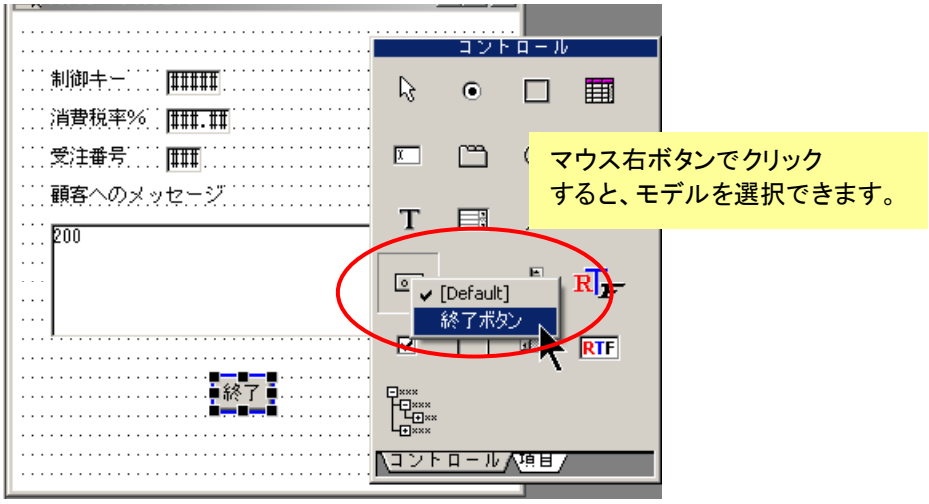


図 7-12 コントロールパレットから登録されているプッシュボタンモデルを選択する

7.3.4. 項目モデルをタスクで利用する

上のプログラムでは、「終了」ボタンにカーソルがパークしませんでしたが、今度はパークするように作ってみましょう。

制御テーブル更新２ の例と同じく、プッシュボタンに対応する変数項目をレコードメインに定義します。ただし、ここでは変数の定義でモデルを参照することによって、詳細な定義をいちいち行わなくても良いようにします。

1. プログラムリポジトリで、制御テーブル更新プログラムをコピーして、「制御テーブル更新 4」と改名します。
2. プログラムを開き、レコードメインを開きます。
3. 最下行の次に新規行を作成し、「セレクト V=変数」にします。
4. その隣の「1」の欄からズームすると、変数項目テーブルが開きます。
5. モデル欄でズームすると、モデル一覧が表示されます。ここに表示されるのは、タイプ＝「F=項目」としたモデルのみです。
6. 最後にある「終了ボタン項目」を選択します。名前、書式などが自動的に設定されます。プロパティシート「ローカル変数特性」を見て、書式、GUI 表示、デフォルト値などが「終了ボタン項目」モデルから継承されていることを確認してください。
7. Enter キーを押して、レコードメインに戻ります。

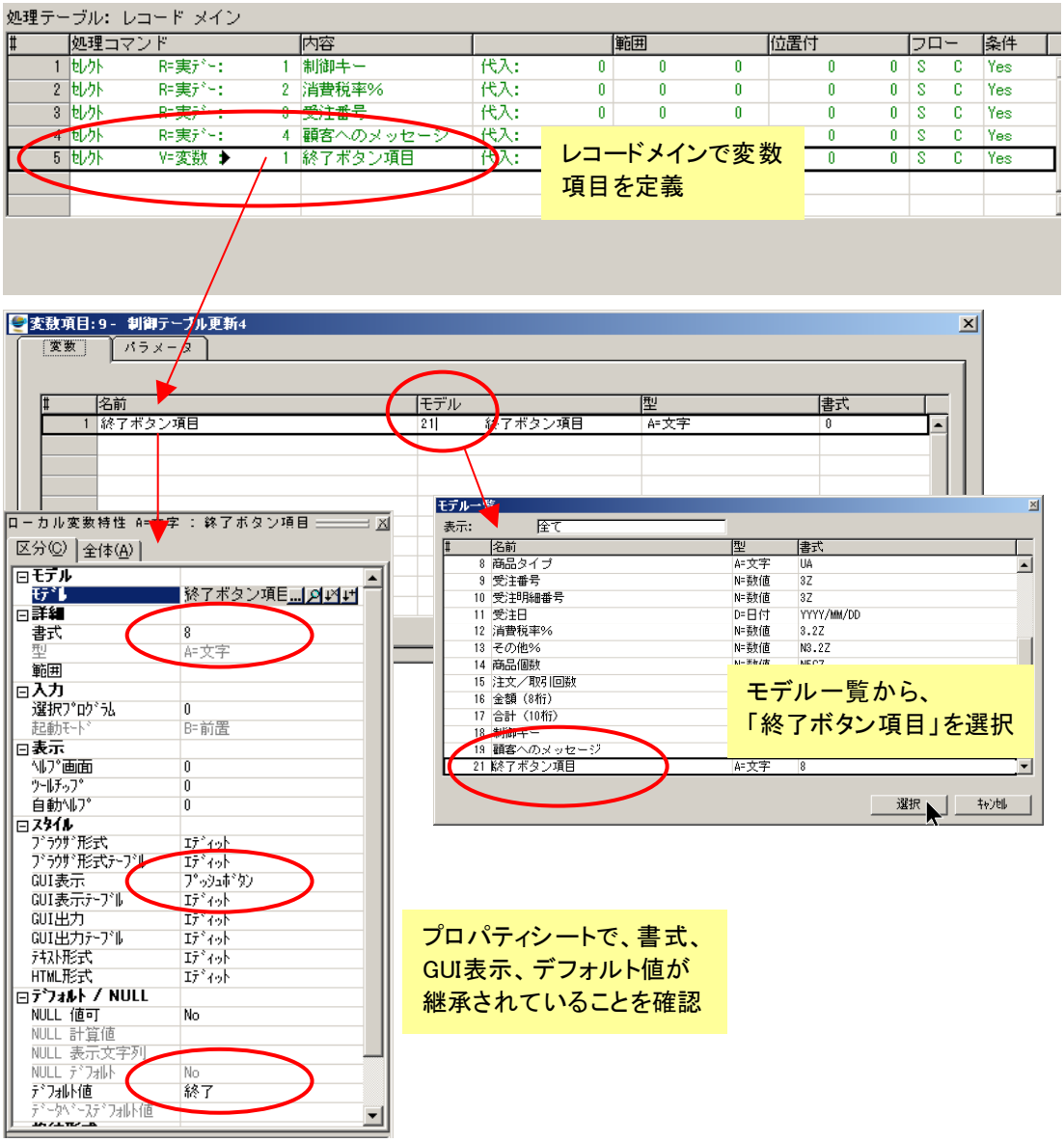


図 7-13 プッシュボタン用の変数項目を定義

8. フォームエディッタを開き、「項目」パレットから「終了ボタン項目」を選び、フォームに配置します。これにより、変数項目 E (終了ボタン項目)の設定がすべて継承されます。コントロール特性で確認してください。
9. フォームエディッタ、タスクを閉じます。

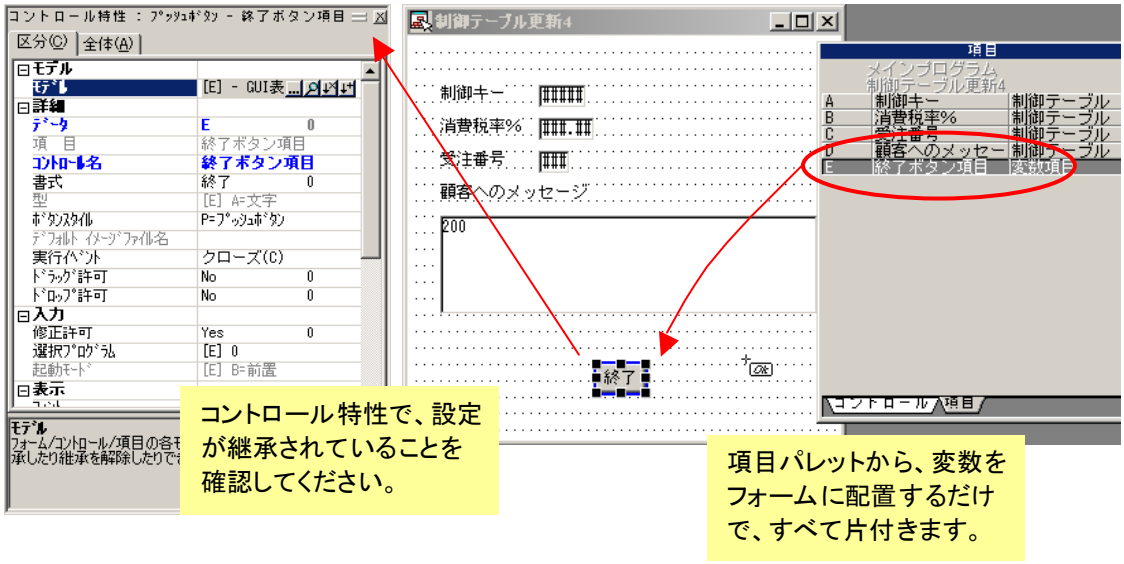


図 7-14 プッシュボタンの変数をフォームに配置

プログラムリポジトリに戻り、実行して、動作を確認してください。今度は「終了」ボタンにカーソルがパークするはずです。もちろん、「終了」ボタンをクリックすれば、プログラムは終了します。

7.4. 本章のまとめ

- Magic プログラムでのプッシュボタンの使いかたの基本を学びました。
- イベントの基本として、現在開いているウィンドウを閉じる「クローズ」という内部イベントを使ってみました。
- プッシュボタンにパークする場合(レコードメインに、対応する変数項目が定義されている場合)と、パークしない場合(フォームにのみ配置されている場合)とを見ました。
- モデルリポジトリに、プッシュボタンモデルと、プッシュボタンに関連づけられたデータ項目モデルとを定義しました。
- モデルを参照するプログラムを作ってみました。

8. 選択プログラム

選択プログラムというのは、文字通り、一覧表示を行って、その中から選択するためのプログラムです。顧客の選択プログラム、商品の選択プログラムなどが、ペットショップデモでは必要になります。
本章では、Magic で選択プログラムをどのように作成するかを見ていきましょう。例として、顧客の選択プログラムを作成します。
図 8-1 は、本章で作成する顧客選択プログラムと、そのテストプログラムです。

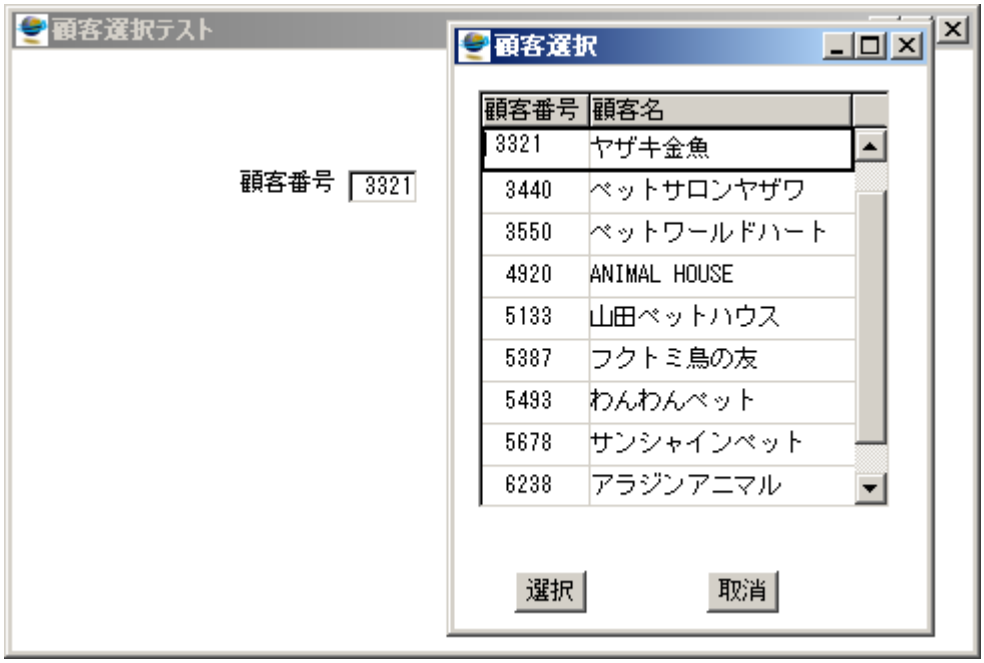


図 8-1 顧客選択プログラムとテストプログラム

参考情報： 本章の内容については、リファレンスマニュアルにより詳細な情報がありますので参考にしてください。

選択テーブルパラメータ	第 6 章 プログラム → タスク → [高度な設定]タブ
パラメータ項目	第 7 章 処理コマンド → セレクト → [セレクト]コマンドの設定欄
パラメータの受け渡し	第 7 章 処理コマンド → コール → パラメータの受け渡し、[パラメータ]テーブル、変数項目とパラメータ項目の違い、Magic でのパラメータの受け渡し方法、など
項目更新コマンド	第 7 章 処理コマンド → 項目更新
式定義	第 6 章 プログラム → 式テーブル
位置付け指定	第 7 章 処理コマンド → セレクト → [セレクト]コマンドの設定欄 → 位置づけ(オプション)

8.1. 選択プログラムの基本形

8.1.1. 表示プログラムの作成

選択プログラムは、基本的に、ライン形式のオンラインプログラムです。まずは、顧客マスターのデータを表示するライン形式のオンラインプログラムを作ってみましょう。ここでは簡単に、APG を使って作ります。

1. プログラムリポジトリで、新規行を作成します。
2. メニュー 「オプション(O)」から「APG(P)」を選び、APG を起動します。(あるいは、Ctrl +G でも同じです)。
3. オプションは「B=照会」、メインテーブルは 2 (顧客マスター)とします。
4. 「選択カラム数」からズームします。「カラム選択」ダイアログが表示されます。
5. 選択プログラムでは、顧客番号と顧客名だけを表示すればよいので、住所以降の「カラム」欄をすべて「0」に設定します。「0」というのは、APG の対象に含めない、ということを意味します。
6. APG ダイアログに戻り、OK ボタンを押すと、プログラムが作成されます。

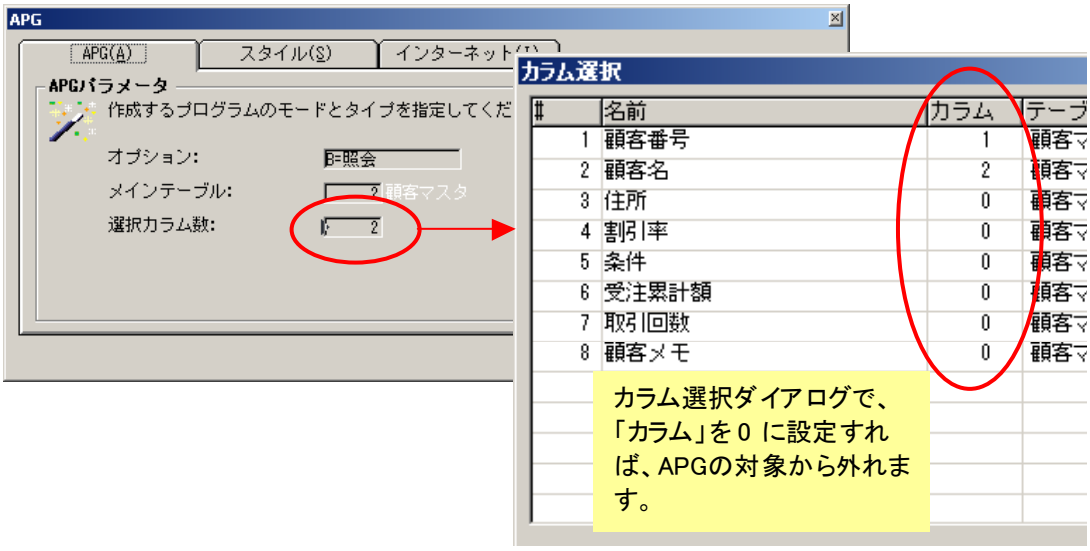


図 8-2 APG で、表示カラムを選択する

これでまずは、顧客マスタのレコードから、顧客番号と顧客名を表示するプログラムができあがりです。

選択プログラムとするために、これに以下の修正を加える必要があります。

- 呼び出したプログラムに、選択された顧客番号を返さなければなりませんので、顧客番号をやりとりするための「パラメータ項目」を定義します。
- ユーザが選択した(具体的には、Enter キーを押した)場合には、その行の顧客番号を、パラメータに設定するようにします。

8.1.2. パラメータ項目の定義

まずは、パラメータ項目を定義します。

1. プログラムを開き、レコードメインを開きます。
2. 先頭に 1 行追加します。(レコードメインのタイトルの部分にカーソルを移動して、F4 キーを押すと、

先頭に新しい行が作成されます。)

3. セレクトコマンドとします。
4. 「R=実データ」は、「P=パラメータ」とします。
5. 次の「1」の欄からズームすると、パラメータテーブルが開きます。パラメータテーブルは、タブが「変数」ではなく「パラメータ」となっている他は、変数テーブルと同じです。
6. 「モデル」欄でズームして、顧客番号モデルを選択します。名前や書式が自動的に設定されます。
7. パラメータであることをわかるように、「P_顧客番号」などといった名前に変更しましょう。
8. レコードメインに戻ります。(Enter キーを押します)

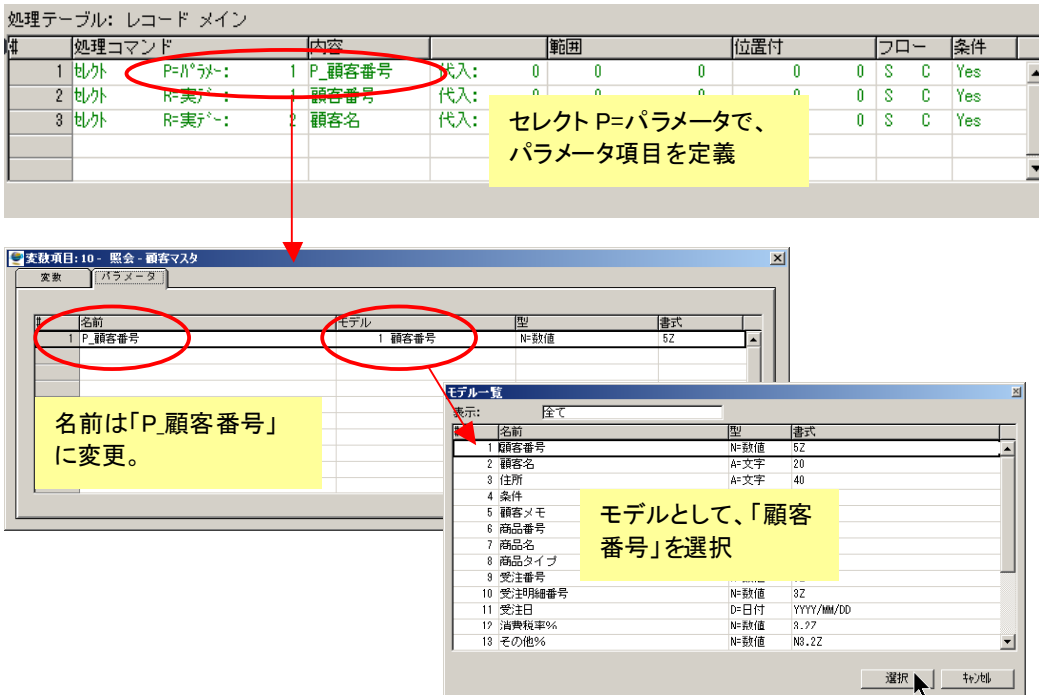


図 8-3 パラメータ項目の定義

8.1.3. Enter キーの処理

次には、ユーザが Enter を押したときに、このパラメータ項目にそのときの顧客番号を設定するロジックを入れないといけません。これを簡単化するために、Magic には「選択テーブル」というパラメータがあります。

「選択テーブル」パラメータは、タスク特性の「高度な設定(A)」タブにあります。(メニュー「編集(E)」から「特性(P)」を選ぶか、あるいは Ctrl + P キーを押して「タスク特性」を開き、「高度な設定(A)」タブを開きます)。

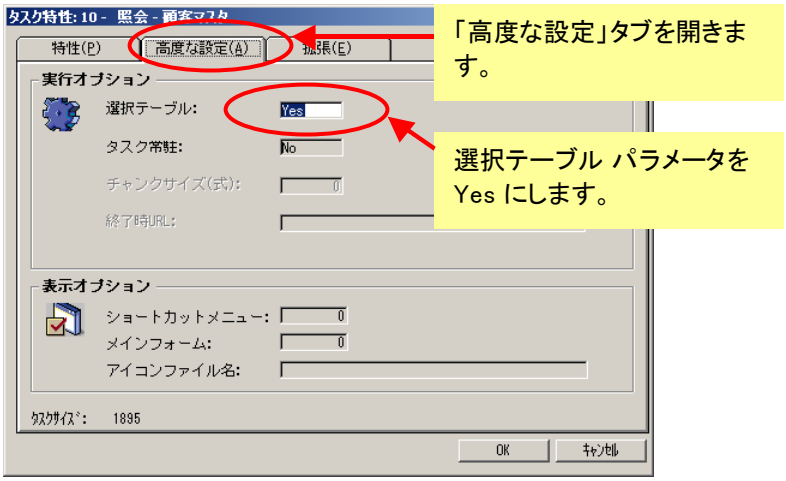


図 8-4 選択テーブルパラメータ

「選択テーブル」のデフォルト値は No ですが、論理型の式、あるいは Yes (True の意味)を設定することができます。

「選択テーブル」パラメータが Yes (または論理値 True)の場合には、実行時に Enter キーが押されたときに、「選択」という内部イベントが発生します。「選択」内部イベントの作用は、以下の通りです。

- レコードメインが終了し、レコード後処理に定義されているコマンドが実行されます。
- その後、タスクは終了します。

従って、今作りたい選択プログラムにするには、次のようにします。

（1）「選択テーブル」パラメータを Yes に設定します

1. タスクを開いた状態で、メニュー「編集(E)」から「特性(P)」を選択します。(あるいは Ctrl + P キーを押します)。タスク特性ダイアログが開きます。
2. 「高度な設定(A)」タブをクリックします。
3. 「選択テーブル」欄に、Y と入力します。設定が Yes に変わります。
4. OK ボタンを押して、ダイアログを閉じます。

（2）レコード後処理に、パラメータ値を更新するコマンドを定義します

1. 「タスク定義」テーブルで、レベルは「R=レコード」、イベントが「S=後処理」の行をクリックします。下半分が「処理テーブル： レコード 後処理」となります。
2. 「処理テーブル：レコード後処理」のタイトル行でクリックします。カーソルが移動します。
3. F4 キーで新規行を作成します。
4. ダブルクリックするとコマンド選択のコンボボックスが現れるので、「U=項目更新」を選びます。(あるいは単に U と入力しても同じです)。
5. 「???」の欄に移動し、ズームします。項目一覧が表示されます。
6. パラメータ変数 A (P_顧客番号)を選択します。
7. 「式：」欄に移動し、ズームします。式テーブルが開きます。
8. 式テーブルでは、次のように式を定義します。

1. F4 で新規行を作成します。
2. 「B」と入力します。B というのは、顧客マスタの顧客番号カラムの実データ項目を表す変数記号です。
3. OK ボタンを押します。「式：」欄に戻り、今作成した式の番号 1 が設定されています。

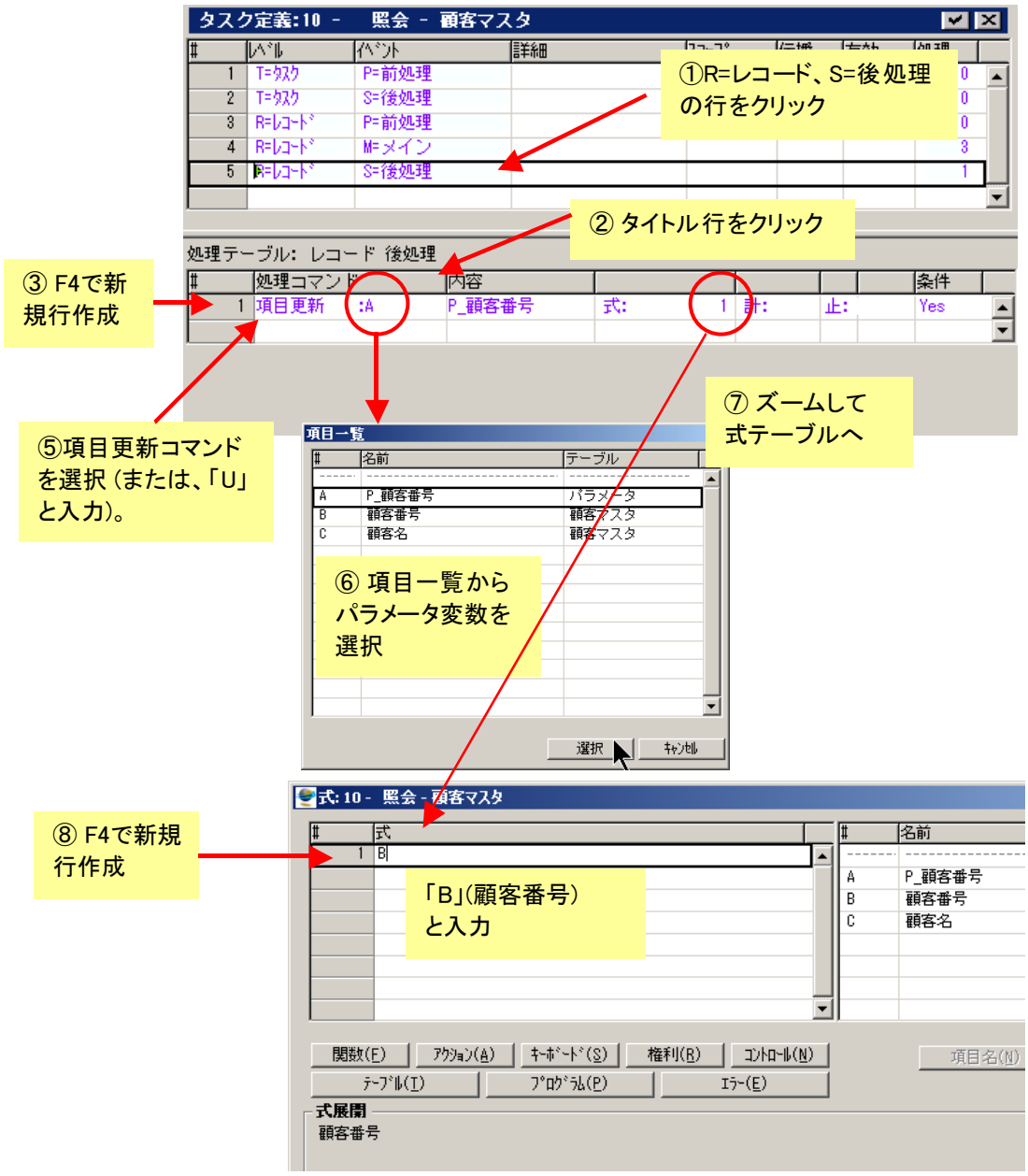


図 8-5 レコード後処理で、パラメータを更新

このように指定した「項目更新」コマンドは、代入文と同じく、式 #1 で計算した値をデータ項目 A に代入するものです。

参考： Magic プログラム内での式定義について

図 8-5 で見たように、Magic のプログラム内で使う計算式は次のように扱います。

- すべての式は、式テーブルに定義します。

- 式の参照は、式テーブルでの式の**番号**により行います。
- 式中で使うデータ項目は、記号 (A より始まり、B、C、・・・、BA、BB、・・・ と続く)により参照します。
- データ項目の記号は、レコードメインにセレクトコマンドで定義された順に、自動的につけられる。

C++、Java、VB などの言語系では、「式テーブル」というものはないし、変数はすべて変数名により参照するので、Magic 流のこのやりかたは最初はちょっと戸惑うかもしれませんが、うまく活用すればキー入力する量を大幅に減らすことができ、プログラムの作成を速く行うこともできますので、うまい使い方を工夫してください。

8.1.4. テストプログラムを作る

顧客番号の選択プログラムを呼び出すテストプログラムを作って、動作を確認してみましょう。
テストプログラムですから、顧客番号を格納する変数がひとつあるだけのプログラムで十分です。この変数でズームし、今作った顧客番号選択プログラムを呼び出し、選択した番号が設定されることを確認するようなものです。

以下の要領でテストプログラムを作ってください。もうすでに習ったことばかりですので、細かな手順は省略します。

1. プログラムリポジトリの最後に新規プログラムを追加します。
2. プログラムを開き、タスク特性で名前を「顧客選択テスト」とします。
3. レコードメインで、セレクトコマンド(タイプは V=変数)により、変数項目を作成します。
4. 変数テーブルにズームし、モデルとして「顧客番号」モデルを選びます。自動的にデータ型は数値、書式は 5Z となります。
5. フォームエディッタを開き、項目パレットからこの変数項目をフォーム上に配置します。

次に、ここから「顧客選択」プログラムを呼び出すにはどうすればよいのでしょうか？プログラムを呼び出すには、一般には「コール」コマンドを使うのですが、コールコマンドの一般的な使い方は後の章で説明し、ここではコールコマンドを使わずに、項目の特性にある「選択プログラム」を使ってみましょう。

「選択プログラム」特性は、データ項目 (実項目、変数項目、パラメータ項目いずれにも共通です)の特性のひとつで、次のようにして見ることができます。

1. 変数テーブルを開きます。(セレクト V=変数 コマンドからズームします。あるいは、Ctrl + V でも開きます)
2. プロパティシートで、ローカル変数特性の中に「選択プログラム」があります。
3. 「選択プログラム」からズームすると、プログラム一覧が表示され、プログラムを選択できます。

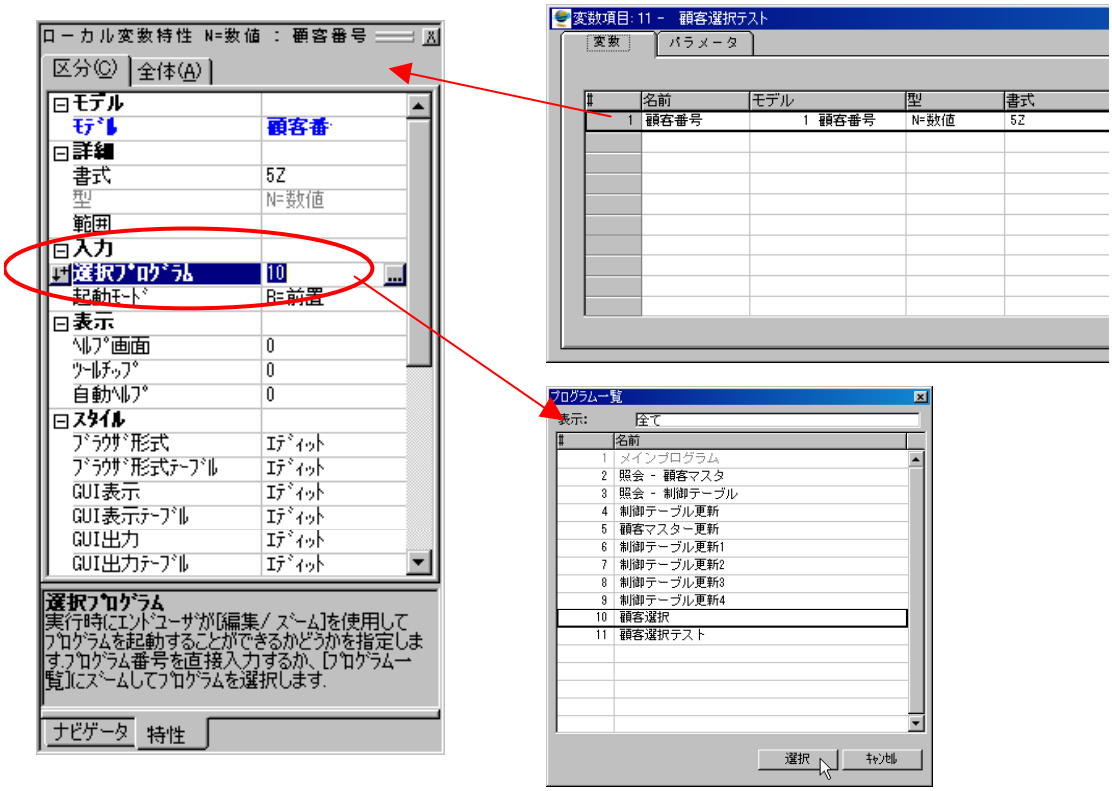


図 8-6 「顧客番号」モデルに、選択プログラムを指定

「選択プログラム」特性にプログラムが設定されていると、次のような効果があります。

- 実行時、その項目にカーソルがパークしているとき、ユーザがズーム(F5 キーまたはマウスのダブルクリック)を行うと、設定されているプログラムが呼び出されます。
- このとき、その項目が呼び出すプログラムへのパラメータとして与えられます。
- プログラムから戻ってきたときには、パラメータの値が、その項目に書き戻されます。

したがって、今作っているテストプログラムでも、変数の「選択プログラム」特性に、先に作った顧客選択プログラムを設定して、実行してみることにしましょう。

実行して動作を確認してみましょう。

1. プログラムリポジトリから、「顧客選択テスト」プログラムを F7 で起動してください。
2. F5 キーを押してください。顧客選択プログラムが開くはずです。
3. 3550 の「ペットワールドハート」にカーソルを移動し、 Enter キーを押してください。顧客選択プログラムが閉じて、項目には 3550 が設定されているはずです。

このようになれば、選択プログラムが正しく動作していることになります。

8.2. 位置づけ機能を加える

さて、上のテストプログラムで、項目の値が 3550 に設定されている状態で、再度 F5 キーを押してみてください。再び顧客選択プログラムが開きますが、カーソルは先頭の行(1008 の千葉ペットショップ)にあります。つまり、前の値が何であっても、選択プログラムは常に先頭の行から表示する、という動作になっています。これは少し不親切で、できれば、初期画面として、パラメータで与えられた値 (3550 のペットワールドハート)に位置づけすべきでしょう。

このために、顧客選択プログラムを改造して、初期状態で、パラメータとして渡された値でレコードの位置づけをするようにしましょう。初期状態でのレコード位置づけのめの条件は、セレクトコマンドの「位置付」欄に式で指定します。

1. 顧客選択プログラムを開きます。
2. レコードメインを開きます。
3. 2 行目の「セレクト R=実データ 1 顧客番号」の行に移動します。
4. 「位置づけ」の最初の欄 (位置付開始)からズームしてください。式テーブルが開きます。
5. 式テーブルで最後に 1 行追加し、A (P_顧客番号) としてください。
6. OK ボタンを押すと、「位置づけ」の最初の欄に、今定義した式の番号 2 が設定されます。
7. プログラムを閉じてください。

このように定義することで、初期画面では、パラメータに与えられた顧客番号を使ってレコードの位置づけがされます。

なお、ここでは、位置付けの「開始」欄だけに式を設定しましたが、これは最小値のみを指定し、最大値は指定していないことを意味します。従ってテーブル中に正確に一致するレコードがなかったとしても、その値よりも大きい最小のレコードを見つけ、そこに位置づけをすることになります。

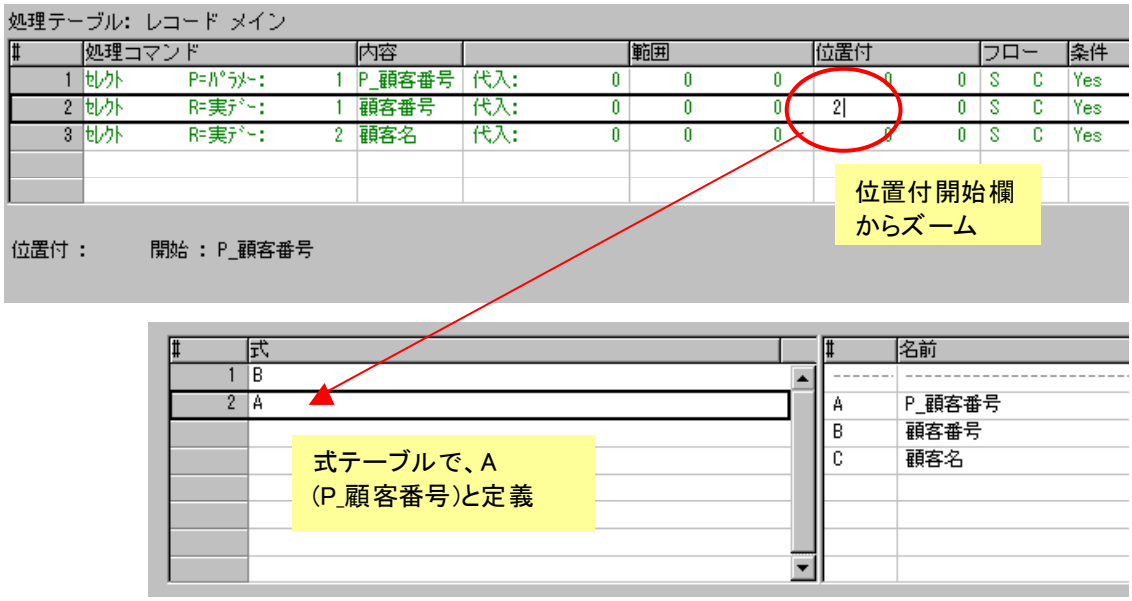


図 8-7 選択プログラムに位置付け機能を追加

再度実行して、位置づけがなされるか確認してみましょう。

1. プログラムリポジトリから、「顧客選択テスト」プログラムを F7 で起動してください。

2. F5 キーを押してください。顧客選択プログラムが開きます。
3. 3550 の「ペットワールドハート」にカーソルを移動し、Enter キーを押してください。顧客選択プログラムが閉じて、項目には 3550 が設定されています。
4. 再度 F5 キーを押してください。顧客選択プログラムが開きます。このとき、3550 のペットワールドハートにレコードが位置づけられていることを確認してください。
5. Enter キーで戻ってください。
6. 今度は項目 5000 と入力して、F5 キーを押してください。顧客選択プログラムが開いたとき、顧客マスタには 5000 というレコードがないので、5000 よりも大きくて最小のレコードである 5133 の山田ペットハウスに位置づけられています。

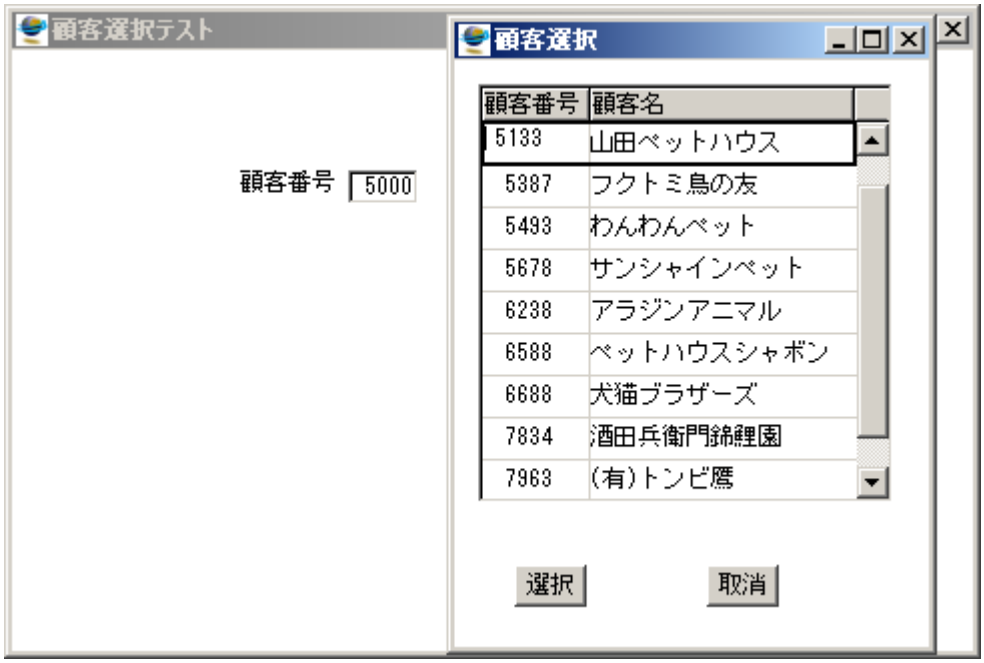


図 8-8 顧客選択プログラムの初期位置づけ

8.3. 「選択」ボタンをつける

選択プログラムは、Enterキーで選択するのも良いですが、プッシュボタンでも選択できるようになっていると便利ですので、「選択」と「取消」のプッシュボタンを追加しましょう。まずは「選択」のボタンをつけてみます。

前に「選択テーブル」のパラメータについて説明したときに、「実行時にEnterキーが押されたときに、『選択』という内部イベントが発生します」と説明しました。選択プログラムでの選択動作は、「選択」という内部イベントによりトリガされるので、プッシュボタンにも「選択」内部イベントを設定してやればよいことになります。

このような「選択」ボタンはよく使うボタンなので、モデルに定義して使いましょう。モデルでの定義のしかたは、「終了」ボタンの場合とまったく同じで、ただ、イベントとして「クローズ」ではなく「選択」を設定し、書式（ボタンのラベル）として「終了」ではなく「選択」とするところだけが異なります。

1. モデルリポジトリを開き、最後に新しいモデルを追加します。
2. 名前は「選択ボタン」、クラスは「D=GUI 表示」、型は「P=プッシュボタン」
3. プロパティシートで、書式は「選択」、実行イベントは内部イベントの「選択」とします。

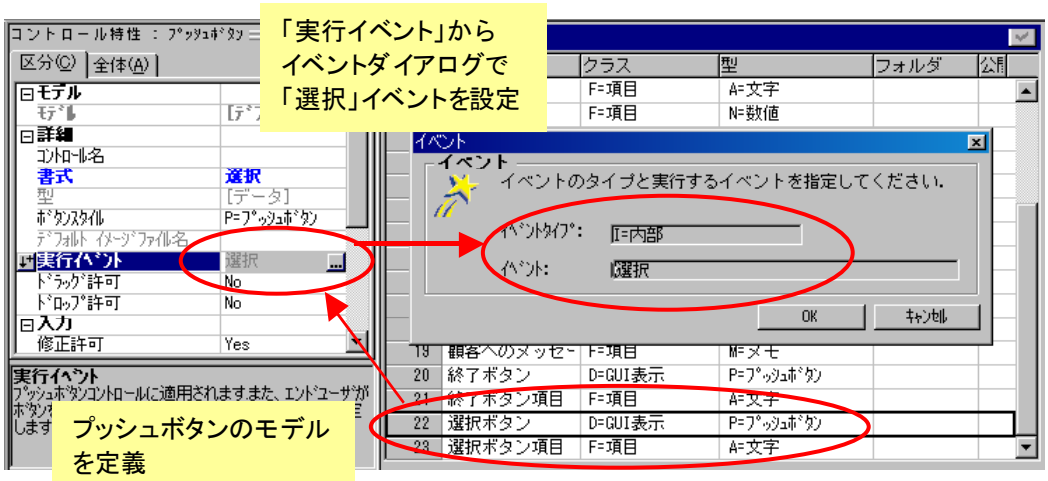


図 8-9 「選択ボタン」モデルの「実行イベント」特性に、「選択」イベントを設定

上の例では、ついでに、この選択ボタンを参照するデータ項目のモデル「選択ボタン項目」も定義しました。これも、「終了ボタン項目」とまったく同じ要領で定義できますので、手順は省略します。

次に、このモデルを使ったプッシュボタンを、フォームに貼り付けましょう。

1. プログラムリポジトリを開き、「顧客選択」プログラムを開きます。
2. フォームエディッタを開きます。
3. テーブルコントロールの高さを少し低くして、下にボタンを配置するための余白を作ります。
4. コントロールパレットのプッシュボタンで、マウスで右クリックすると、定義されているプッシュボタンモデルの一覧が表示されるので、「選択ボタン」をクリックします。マウスカーソルの形状が変わります。
5. そのまま、ボタンを配置したところでクリックします。選択ボタンが配置されます。

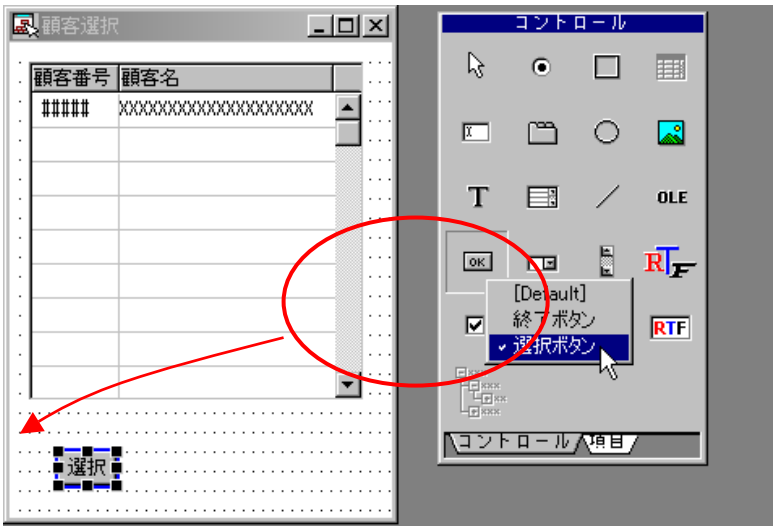


図 8-10 コントロールパレットから、プッシュボタンモデルを選択して配置

8.4. 「取消」ボタンをつける

「取消」ボタンというのは、選択をせずに、すなわち、渡されたパラメータの値は変えずに、そのまま選択プログラムを終了するためのものです。これは何のことはない、「終了」ボタンと同じく、「クローズ」内部イベントを発行するプッシュボタンを使うだけでよいのです。

ただし、ボタンのラベルは「取消」のほうがわかりやすいので、「終了」とは別に、これもモデルを定義しましょう。

モデル定義の手順は「終了ボタン」や「選択ボタン」の場合とまったく同じです。ただし、書式は「取消」とし、実行イベントは「クローズ(C)」内部イベントにします。

ついでに、「取消ボタン」に関連づけられたデータ項目モデル「取消ボタン項目」モデルも作っておきましょう。

図 8-11 「取消」ボタンモデルの登録

このモデルを使ったプッシュボタンを、「選択」ボタンの場合と同じ要領で、フォームに配置しましょう。

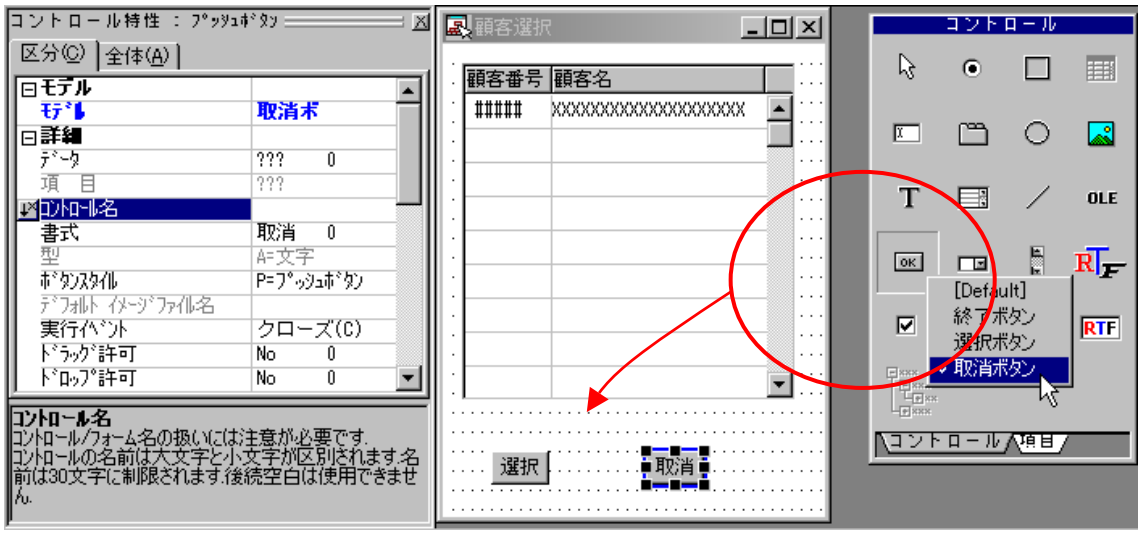


図 8-12 取消ボタンモデルを選択して、フォームに配置

以上で選択プログラムが完成です。

テストプログラムを実行して、動作を確認してください。

8.5. 顧客番号モデルに選択プログラムを設定する

ここで作った選択プログラムは、アプリケーションのどこからでも、顧客番号のあるところでは使えるようになっていると便利です。そのために、最後の仕上げとして、モデルリポジトリの「顧客番号」モデルの「選択プログラム」特性に、顧客選択プログラムを設定しましょう。

モデルリポジトリを開き、「顧客番号」モデルのプロパティシートで「選択プログラム」特性を 10 (顧客選択プログラムの番号)に設定するだけです (図 8-13)。

これだけで、アプリケーション中、このモデルを参照している項目ではすべて、ズームをして選択プログラムを開くことができるようになります。

Magic のモデルが、単なるデータタイプ定義だけではない、強力な機能を持っていることを示すひとつの例です。

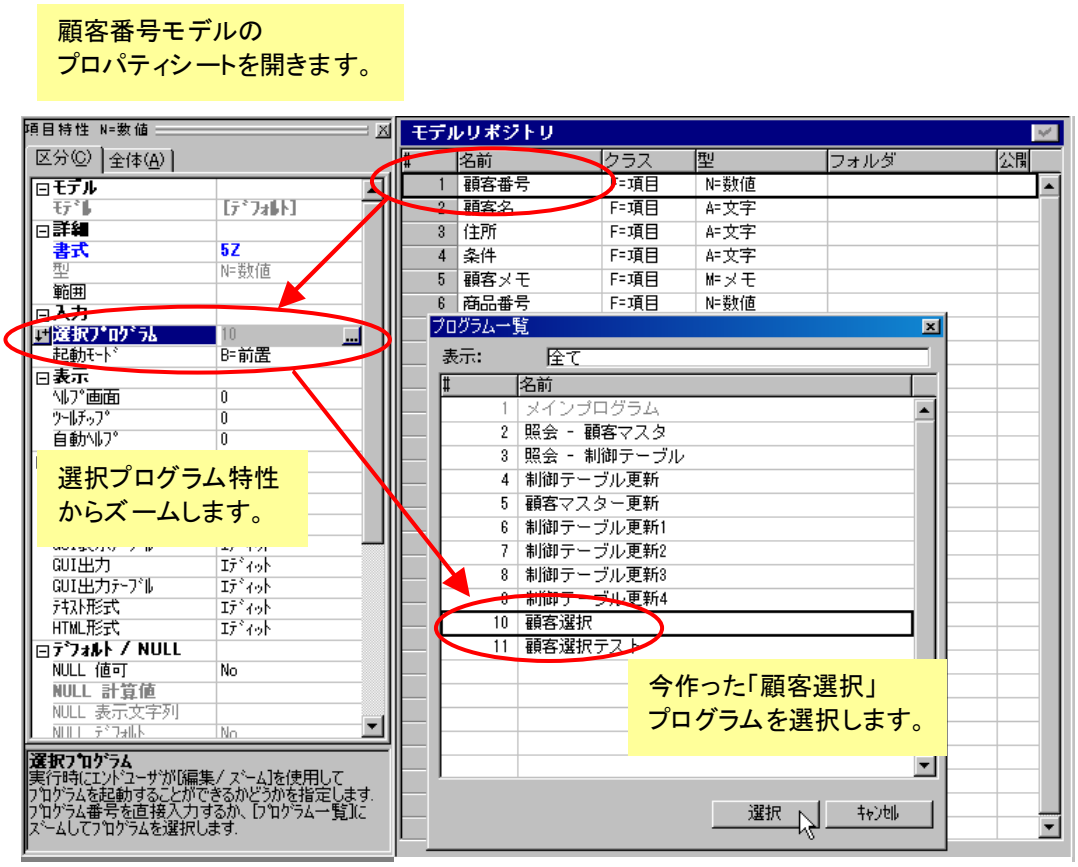


図 8-13 「顧客番号」モデルに、選択プログラムを設定

8.6. 課題

以上、顧客選択プログラムを作成しました。ここで、同様にして、商品選択プログラムとそのテストプログラムも作成してみましょう（図 8-14）。

図 8-14 商品選択プログラムとテストプログラム

商品選択プログラムを作成したら、顧客選択の場合と同様、モデルリポジトリの「商品番号」モデルに、選択プログラムとして登録しておいてください。これは後で使うことになります。

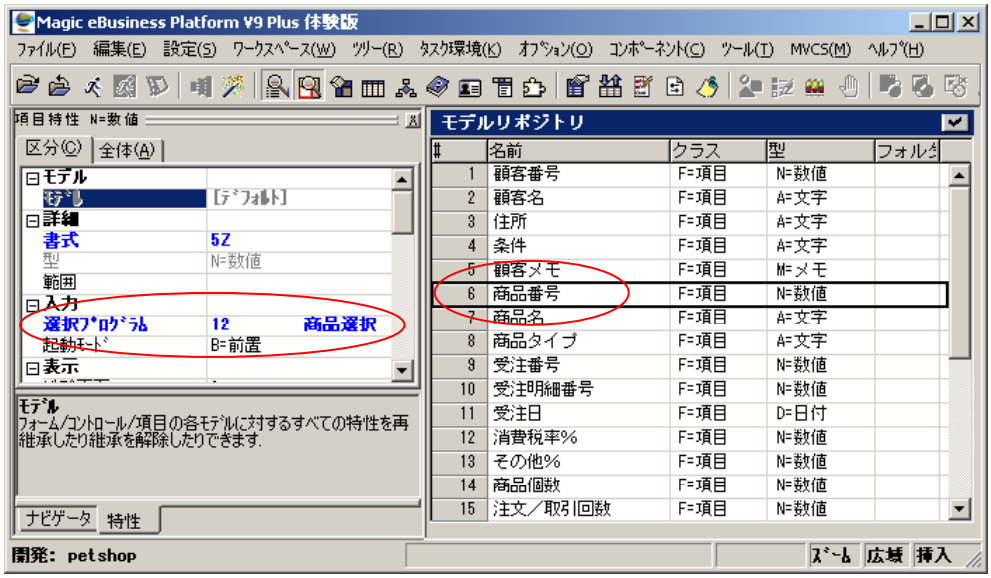


図 8-15 商品番号モデルに、商品選択プログラムを選択プログラムとして登録

9. 親子タスク１．基本構造

今までの例では、ひとつのタスクにテーブルをひとつだけ表示するものばかりでした。実際のアプリケーションでは、複数のテーブルから引き出してきたデータを組み合わせて同時に表示するのプログラムが多くあります。

この章では、そのようなプログラムの基本として、受注入力プログラムを例にとり、

- データリンク
- １：N の親子タスク

という二つの方法で複数のテーブルのレコードを組み合わてプログラムを作成する方法を説明していきます。

図 9-1 は、これから作る受注入力プログラムの完成形です。

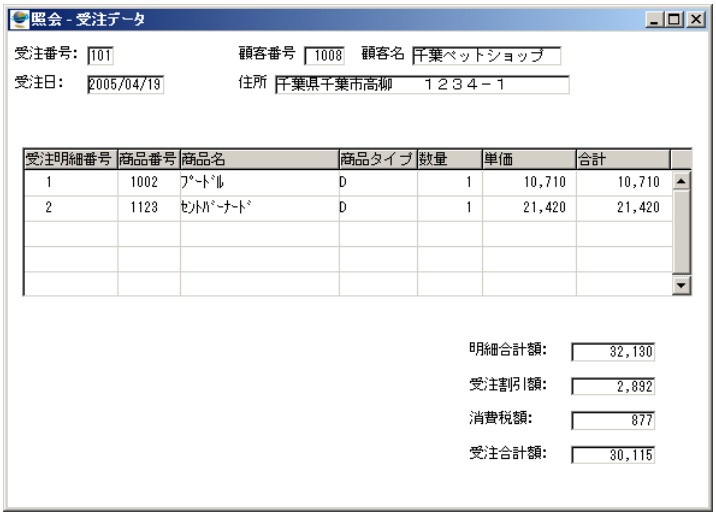


図 9-1 受注入力プログラムの完成形

このようなデータ構造は １：N 関係と呼ばれます。ひとつの受注レコードに対し、複数の受注明細レコードがあるからです。このようなデータ構造を表示・操作するには、親子タスクを定義し、「１」の方のレコード(この場合、受注データ)を親タスクが表示し、「N」の方のレコード(受注明細レコード)は子タスクが表示する、というようにするのが普通です。

参考情報： 本章の内容については、リファレンスマニュアルにより詳細な情報がありますので参考にしてください。

リンクコマンド	第 7 章 処理コマンド → リンク
範囲指定	第 7 章 処理コマンド → セレクト → [セレクトコマンド]の設定欄 → 範囲(オプション)
フォームの調整	第 6 章 プログラム → フォームテーブル → フォームの操作
タスク終了条件	第 6 章 プログラム → タスク制御 → [タスク制御]ダイアログ
Level 関数	第 8 章 関数 → 関数の説明(アルファベット順) → Level
カーソルのパーク	第 7 章 処理コマンド → セレクト → [セレクトコマンド]の設定欄 → 条件

9.1. 親タスクの作成

最初に、受注データを表示する親タスクを作りましょう。受注データ(受注番号、受注日、顧客番号など)は、1 画面で 1 レコードを表示するので、スクリーンモードになります。

9.1.1. APG

プログラムリポジトリで新規プログラムを作成し、APG と起動して、スクリーンモードで「受注データ」テーブル(テーブル 4 番)を表示するプログラムを作りましょう。

APG でのプログラムの作り方は、すでに出てきたので、ここでは具体的な操作方法を省略します。

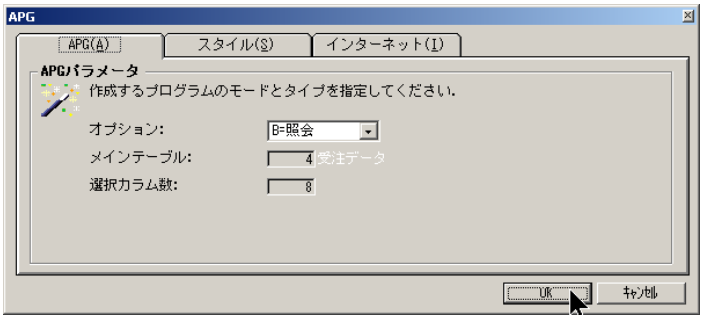


図 9-2 APG ダイアログ

このようにして作成したプログラムを実行すると下図のようになります。

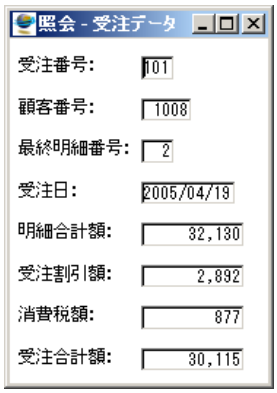


図 9-3 APG で作成した直後の画面

最終形を頭に描きながら、プログラムを開き、フォームエディッタでフォームを修正してください。だいたい、図 9-4 のような感じにします。

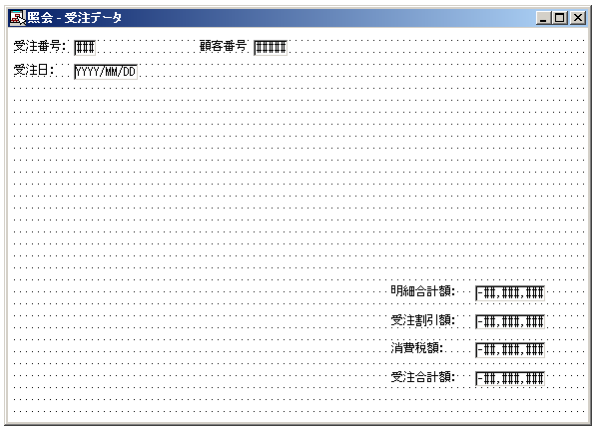


図 9-4 フォームエディッタで修正

これを実行すると、次のようになります。

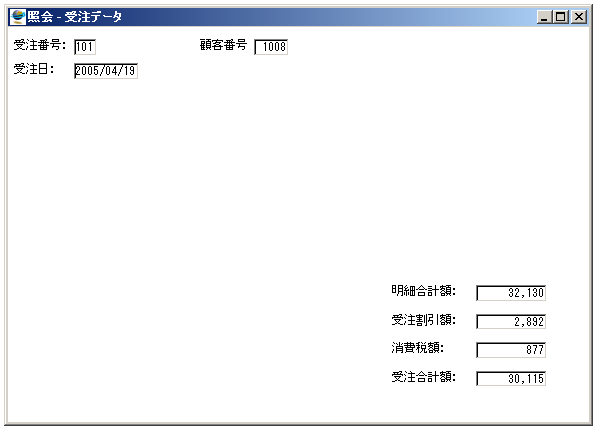


図 9-5 親タスクの実行画面

9.1.2. 顧客マスタをリンク

さて、この画面では、顧客についての情報が顧客番号しか表示されていません。これは、「受注データ」テーブルには、顧客番号しか格納されていないためですが、これでは見るときに不便なので、顧客の名前と住所も表示する必要があります。

顧客の名前と住所とは、顧客マスターテーブルに格納されています。したがって、受注データレコード中の顧客番号を使って、顧客マスターテーブルのレコードを引き出し、データビューに取り込んでやる必要があります。このように、別テーブルのレコードを、決まった条件によって読み込んでくる処理を、Magic ではデータリンクと言います。これは、リレーショナルデータベースの外部結合(Outer Join)の処理に似ています。

顧客番号を使って、顧客マスターのレコードを読み込んでくるデータリンクは、次のようにして定義します。

1. レコードメインを開きます。
2. 「セレクト R=実データ 2 顧客番号」の行の下に、F4 で 1 行作成します。
3. 行の先頭をダブルクリックすると、コンボボックスが現れるので、「L=リンク」を選択します。(あるいは、単に L キーを押すだけでも同じです)

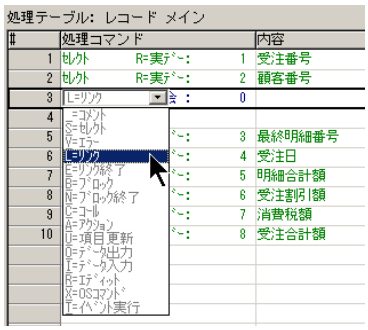


図 9-6 リンクコマンドの作成

4. 「Q=照会」は「リンク方式」と呼ばれるパラメータですが、ここでは Q=照会のままにします。
5. 次のカラム(初期状態は「0」)では、リンクするテーブルを番号で指定します。ここでズームすると、テーブル一覧が出てくるので、これから顧客マスタを選択します。 選択 ボタンを押すと、リンクコマンドに戻り、テーブル番号 2 が設定されます。

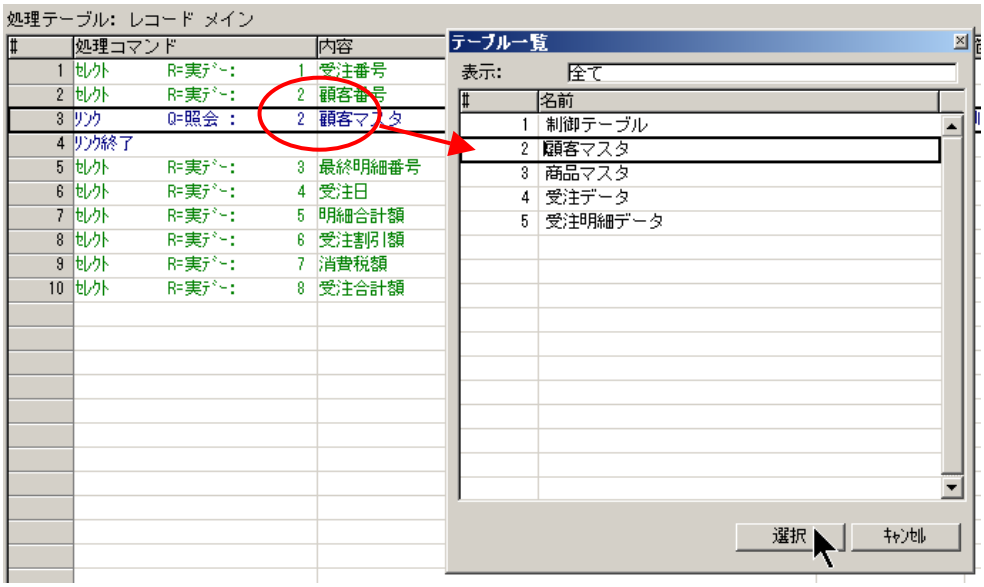


図 9-7 リンクテーブルの設定

6. 「インデックス」欄は、リンク時に使われるインデックスを指定します。ここでは顧客番号を条件としてリンクするので、顧客番号のインデックス(インデックス 1)を選択します。ズームするとインデックス一覧が表示されるので、ここから選択します。
   選択すると、インデックス欄にインデックス番号 1 が設定されると共に、インデックスのセグメント（この場合、顧客マスタの顧客番号カラム)が自動的にセレクトコマンドで定義されます。

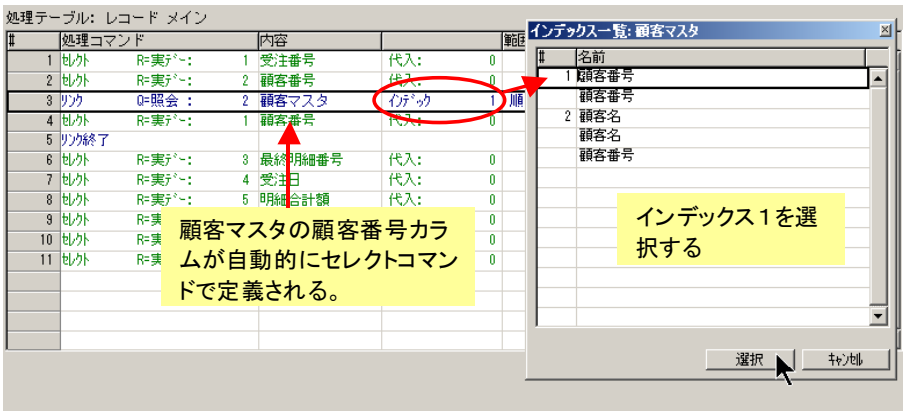


図 9-8 インデックスの選択

7. リンクの条件は、リンクされるほうのレコードのセレクトコマンドの「位置付」欄に、式で指定します。ここでは、顧客マスタの顧客番号が、受注データの顧客番号と等しい、という条件なので、顧客マスタの顧客番号を定義しているセレクトコマンドの、「位置付け」欄に条件を指定します。

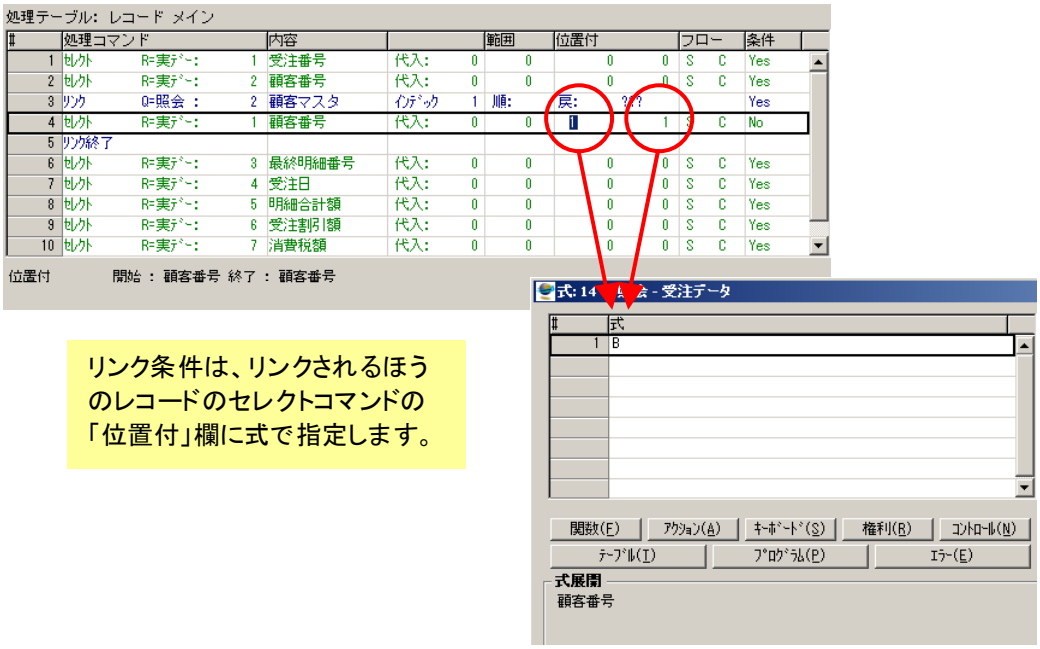


図 9-9 リンク条件の指定

8. 顧客マスタのリンクしたレコードから、顧客名と住所とを引いてきます。これは、「リンク」と「リンク終了」コマンドの間で、セレクトコマンドで定義します。

図 9-10 リンクレコードから、顧客名と住所のカラムも選択

以上で、顧客マスタへのリンクの定義は完了です。

9.1.3. 顧客情報をフォームに配置

今定義した、顧客マスタからの顧客情報(顧客名と住所)をフォームに配置します。配置の方法は、前と同じく、項目パレットから、クリック → フォーム上にドロップ、という形で行います。

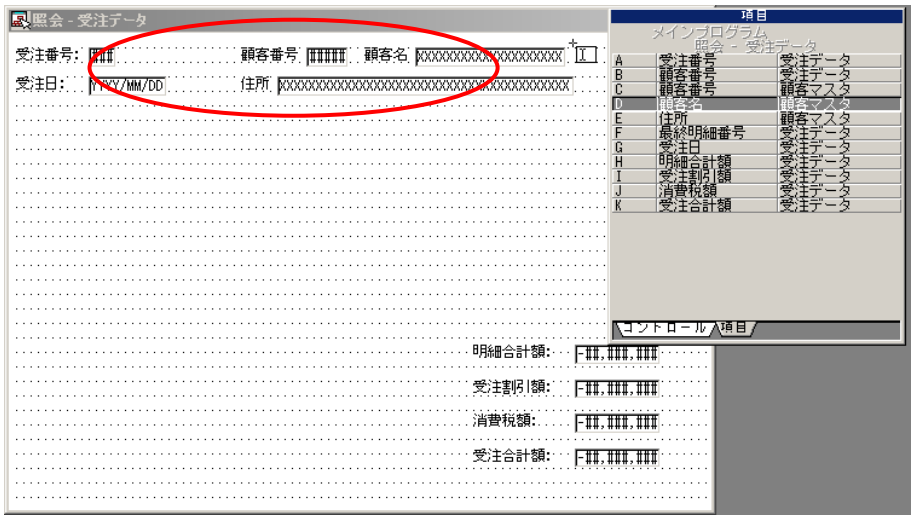


図 9-11 顧客名と住所をフォームに配置

ここで、再度実行してみましょう。

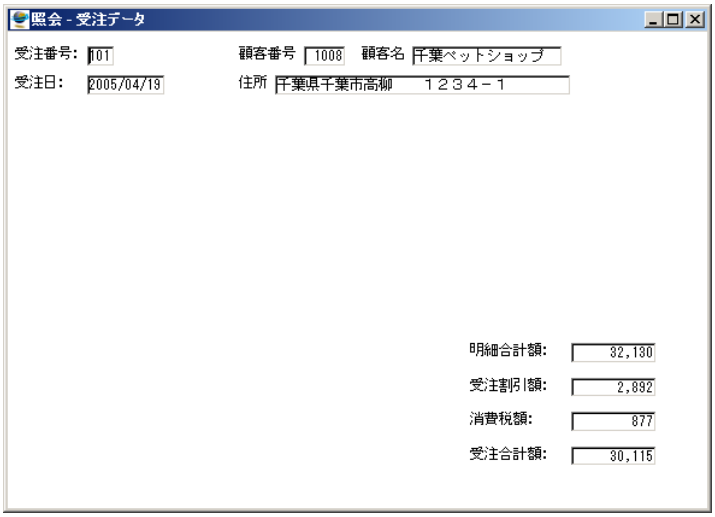


図 9-12 リンク項目も追加した実行画面

顧客番号に対応した顧客名と住所とが表示されるはずです。

PgUp および PgDn キーで別レコードに移り、顧客名と住所とが顧客番号に合わせて変わることを確認してください。

9.2. 子タスク

次に子タスクを作りましょう。子タスクは、明細行をライン形式で表示するものです。

9.2.1. 子タスクを作成する

子タスクは、「ナビゲータ」から作成します。

まずナビゲータを開きます。ナビゲータは、次のいずれかの方法により開きます。

- メニュー「ワークスペース(W)」から「ナビゲータ(N)」を選択する。
- ツールバーから「ナビゲータ」アイコンをクリックする。
- Alt + F1 キーを押す。

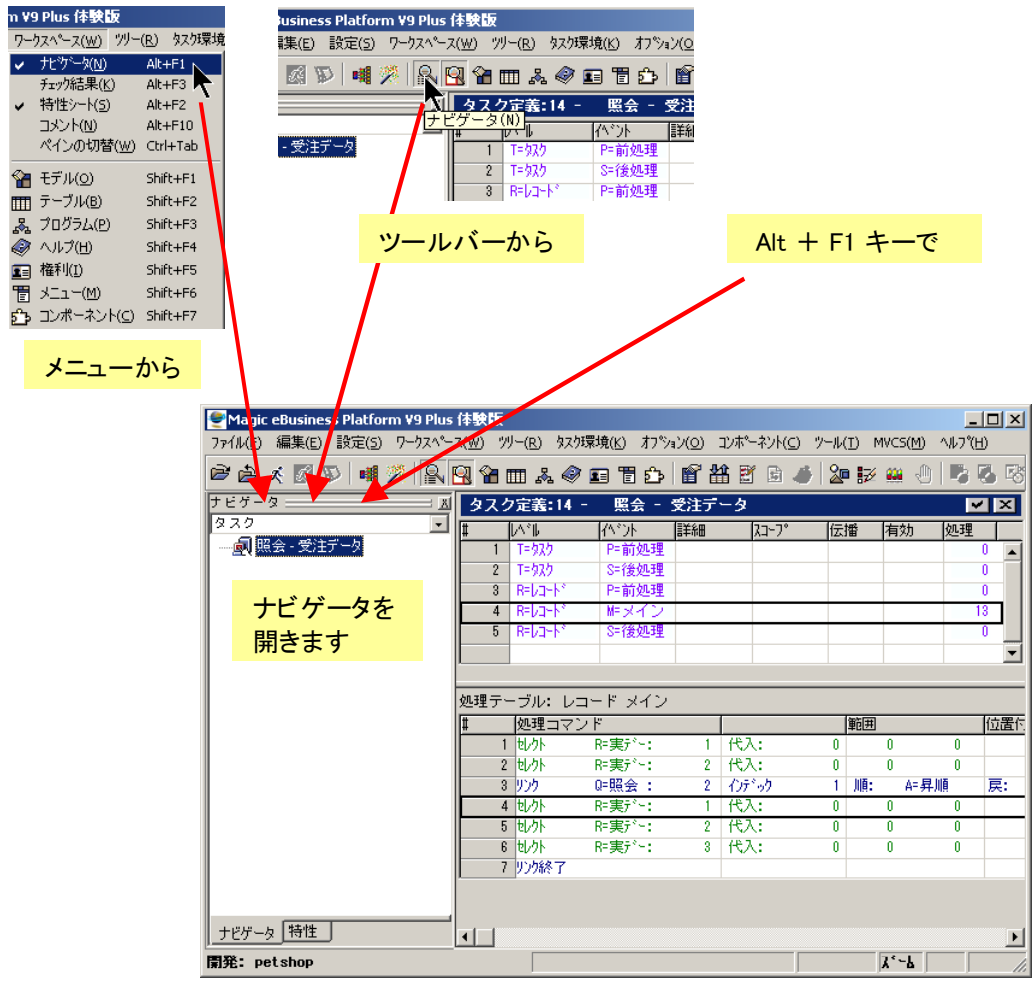


図 9-13 ナビゲータ

ナビゲータを開いた後は、次のようにして子タスクを作ります。

1. ナビゲータ上で、親タスクをクリックする。

2. マウスで右ボタンをクリックし、「行作成(R)」を選ぶ。（単に F4 キーを押しても同じです。）

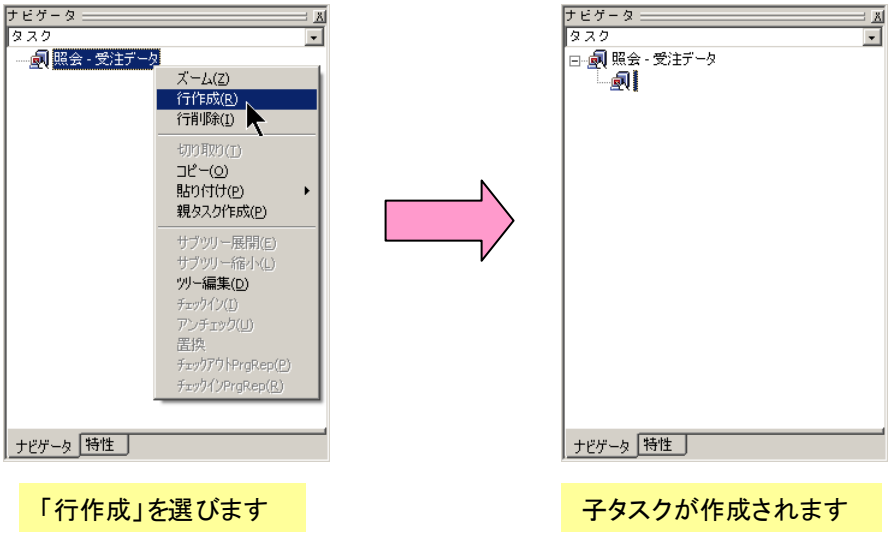


図 9-14 ナビゲータで子タスクを作成

子タスクを作成すると、自動的に子タスクが開き、「タスク特性」ダイアログが開くきますが、次に APG を行うので、ここではとりあえずそのまま「OK」を押して閉じます。

9.2.2. 子タスクへの APG

子タスクは、受注明細データをライン形式で表示するものなので、手っ取り早く、APG でプログラムの原型を作りましょう。

子タスクで APG を行うには、次のようにします。

1. ナビゲータで、子タスクをクリックします。
2. APG を起動します(メニュー「オプション(O)」から「APG(G)」を選択、あるいは、Ctrl ＋ G キーを押す)。
3. メインテーブルとして、ズームしてテーブル一覧から「受注明細データ」テーブルを選択します。
4. OK ボタンを押すと、タスクが作成されます。

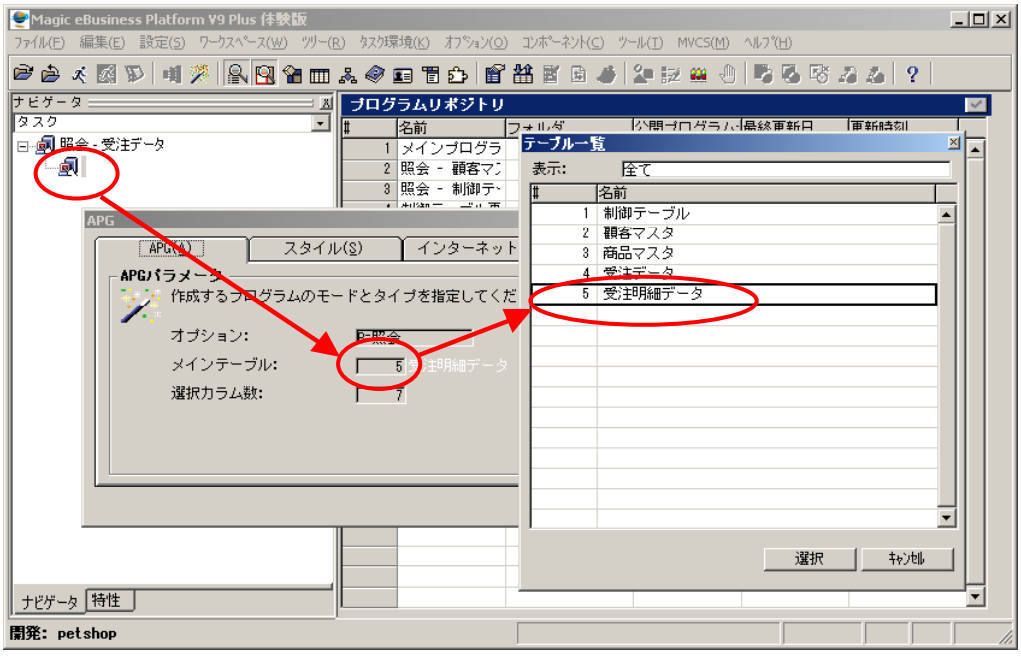


図 9-15 子タスクでの APG

9.2.3. 商品名のリンク

APG の直後では、商品番号は表示されますが商品名は表示されません。これではエンドユーザに不便なので、先の顧客の場合と同じく、商品番号で商品マスターをリンクし、商品名を取得するようにします。
定義の操作は、顧客の場合と同じです：

1. 子タスクのレコードメインを開きます。
2. 「セレクト R=実データ 3 商品番号」の行の直後に、 F4 で新規行を作成します。
3. 「リンク」コマンドを選びます。
4. リンクテーブルは、3 (商品マスタ)で、インデックスは 1 (商品番号)です。
5. リンクコマンドの直後に「セレクト R=実データ 1 商品番号」の行が自動作成されるので、その「位置付」の開始欄(左側)からズームします。式テーブルが開きます。
6. F4 で行を追加し、N (受注明細データの商品番号)とします。
7. OK を押すと、セレクトコマンドの位置付開始欄に戻ります。今定義した式の番号 1 が設定されています。
8. 位置付の終了欄(右側)も 式番号 1 を設定します。

F4 でもう一行作成し、「セレクト R=実データ 2 商品名」とします。

9.2.4. 受注番号の範囲指定

このままでは、受注明細データ全体が子タスクで表示されてしまいますので、親タスクで現在表示している受注番号に関連する受注明細データのみに表示対象を絞る必要があります。
表示対象を絞り込むには、条件付けするデータ項目(今の場合、受注番号)の「セレクト」コマンドで、「範囲」の開始と終了とを指定します。

1. レコードメインを開きます。
2. 第一行目にある「セレクト R=実データ 1 受注番号」の行で、「範囲」の開始欄(左側の欄)にカーソルを移動します。
3. ズームすると、式テーブルが開きます。
4. 1 行追加し、親タスクの「受注番号」の項目 A を式に指定します。
5. OK ボタンを押すと、レコードメインに戻り、範囲の開始欄には、今作成した式の番号 2 が設定されます。
6. 範囲の終了欄も、同じく、式番号 2 を指定します。

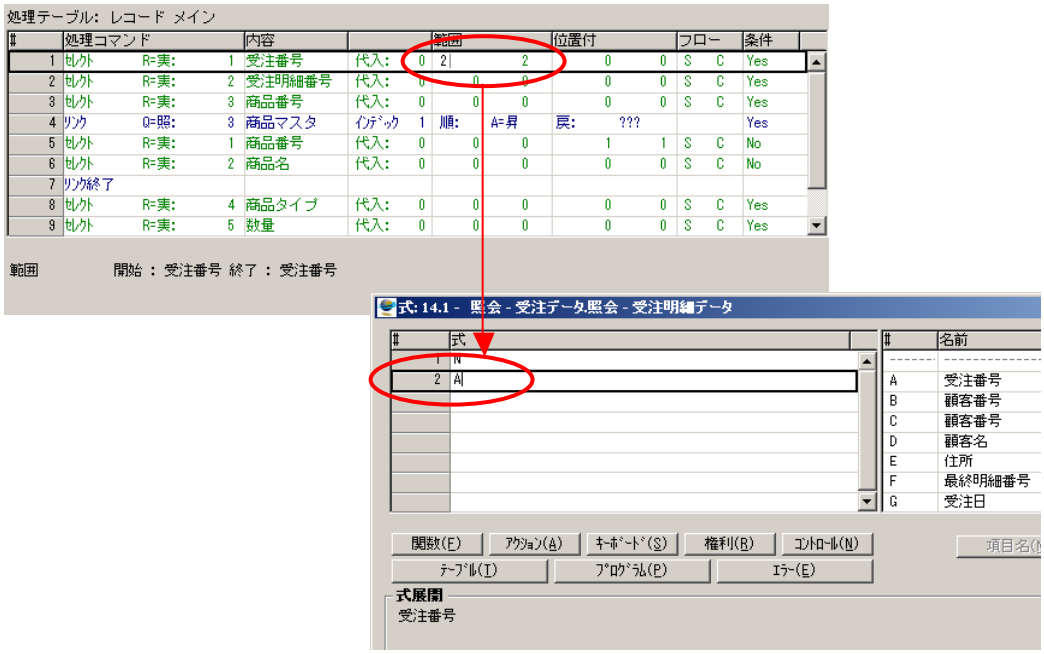


図 9-16 受注番号で範囲指定

9.2.5. フォームの調整

この状態で、子タスクのフォームを開いてみると、下図のようになっています。

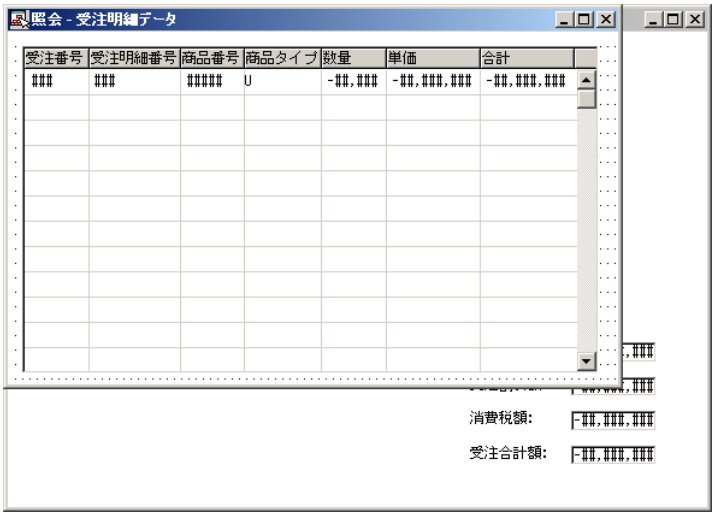


図 9-17 子タスクのフォームの初期状態

このフォームには、次の修正を加える必要があります。

① テーブルコントロールの行数が多すぎるので、5 行くらいにしておきます。それに伴い、子タスクのウィンドウの高さも小さくします。

② 受注番号は、親タスクで表示されているはずで、子タスクでも表示する必要がないので、削除します。

③ 商品名が表示されていないので、追加する必要があります。

④ 子タスクのフォームが親タスクのフォームに重なっていて、このままでは親タスクのフォーム(受注レコードの内容)が見れません。そこで、親子両方のタスクのフォームが具合良く見れるよう、フォームを調整する必要があります。また、子タスクのフォームにはタイトルバーやボーダが不要です。

以下にひとつづつ修正をしていきましょう。

① テーブルコントロールの行数の調整

テーブルコントロールの行数は、テーブルコントロールの高さを調整することで自動的に調整されます。

テーブルコントロールの高さは、次のように行います。

1. フォームエディッタを開きます。
2. テーブルコントロール上をマウスクリックします。テーブルコントロール全体が選択されます。
3. 下辺の ■ をマウスドラッグすれば、高さを調整できます。5 行くらいにしてください。
4. フォームの下に余白ができるので、ウィンドウサイズを調整してください。

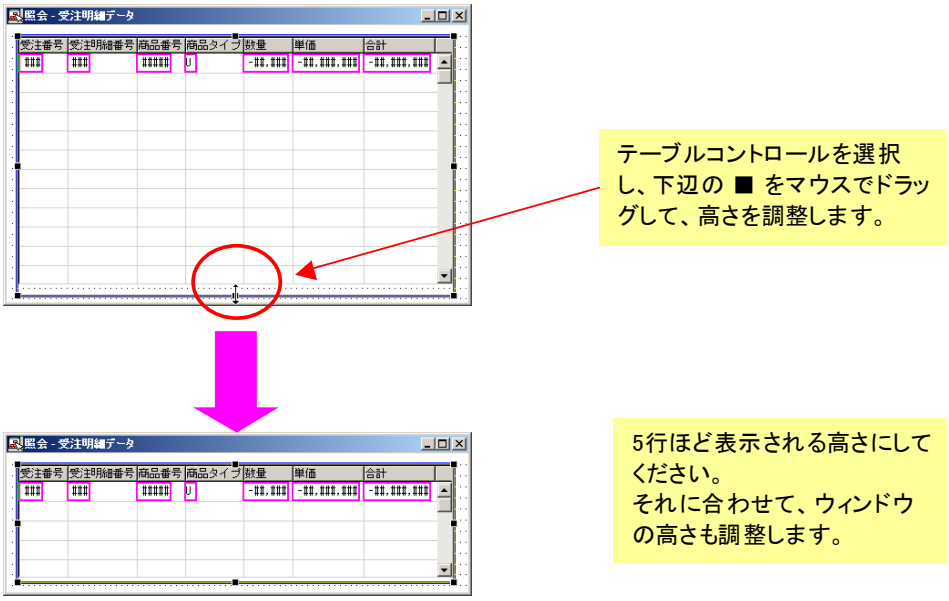


図 9-18 テーブルコントロールと子ウィンドウの高さの調整

② 受注番号の削除

テーブルコントロールから受注番号を削除するには、受注番号のエディットコントロールを削除します。

1. 受注番号のエディットコントロールをクリックし選択します。
2. Delete キーを押します。エディットコントロールが削除されると共に、受注番号のカラム全体が削除され、テーブルコントロールのサイズが自動的に調整されます。

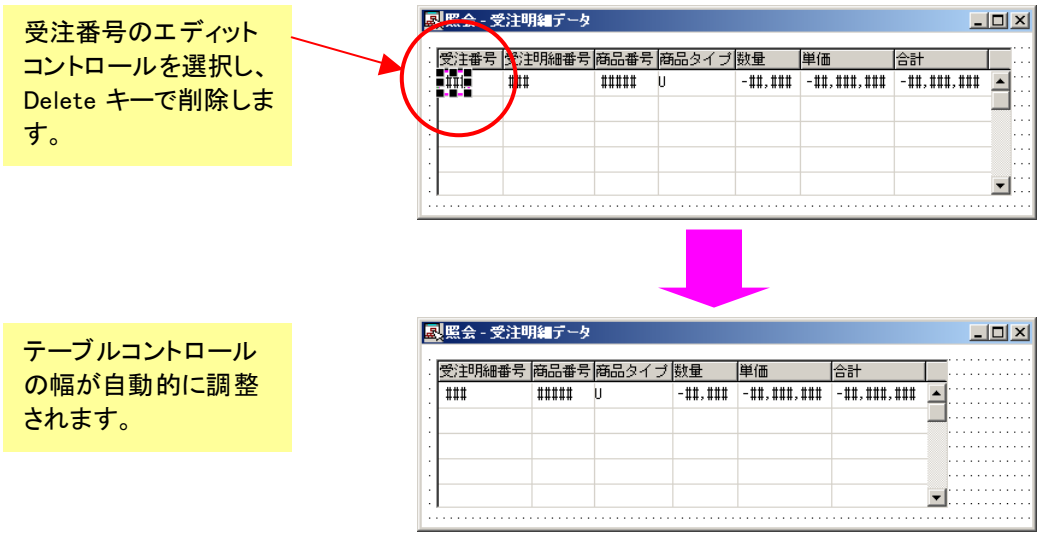


図 9-19 受注番号の削除

③ 商品名の追加

フォームエディッタを開いた状態で、商品名を追加します。

1. 「項目」パレットから、「P 商品名」をクリックします。
2. 「商品番号」のカラム上にマウスを持っていくと、カラムがハイライトされます。この状態でマウスクリックします。商品番号カラムの後ろに、商品名カラムが追加されます。
3. テーブルサイズも自動的に拡大されるので、フォームの幅を調整して全体が収まるようにしてください。

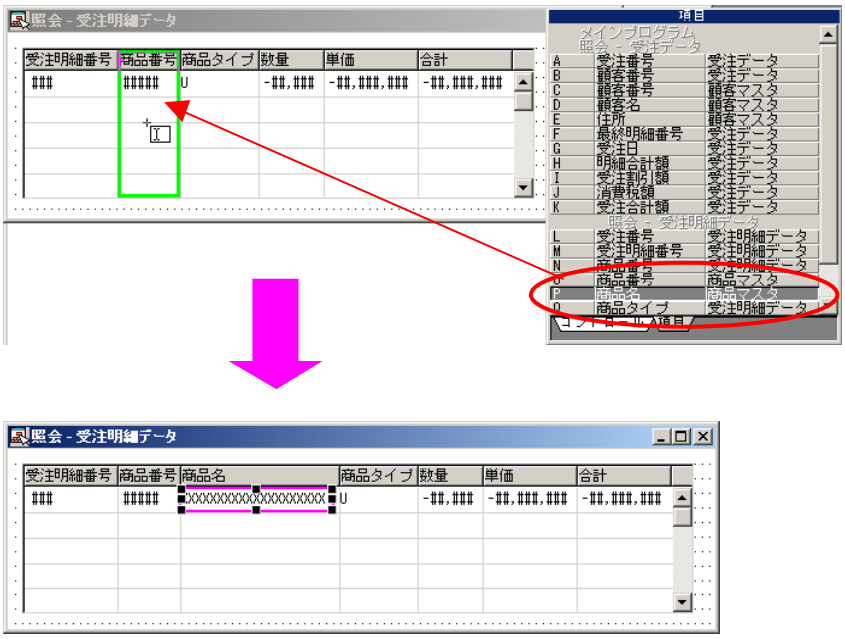


図 9-20 商品名の追加

9.2.6. 「子ウィンドウ」特性の設定

子タスクのフォームの「子ウィンドウ」特性を Yes にします。「子ウィンドウ」特性が Yes の場合には、子タスクのウィンドウは親タスクのウィンドウに依存する形で位置が決められます。つまり、子タスクのウィン

ドウの座標値は親タスクの座標値をもとにして決められ、また、子タスクは親タスクのウィンドウの中にだけ表示されます。このため、例えば、ユーザがタイトルをドラッグするなどして親タスクの位置を移動させたら、子タスクも一緒に移動します。また、子タスクのウィンドウが親タスクのウィンドウに納まりきらない場合には、はみ出た部分は見えなくなります。

「子ウィンドウ」特性が No の場合(デフォルト)では、親のウィンドウと子のウィンドウは独立したウィンドウとして表示されます。

ここの例では、親子が一緒になっている方が良いので、「子ウィンドウ」特性を Yes に設定します。

このとき一緒に、タイトルバーやボーダを表示しないようにするため、「タイトルバー」特性を No とし、「境界」特性は「N=なし」とします。

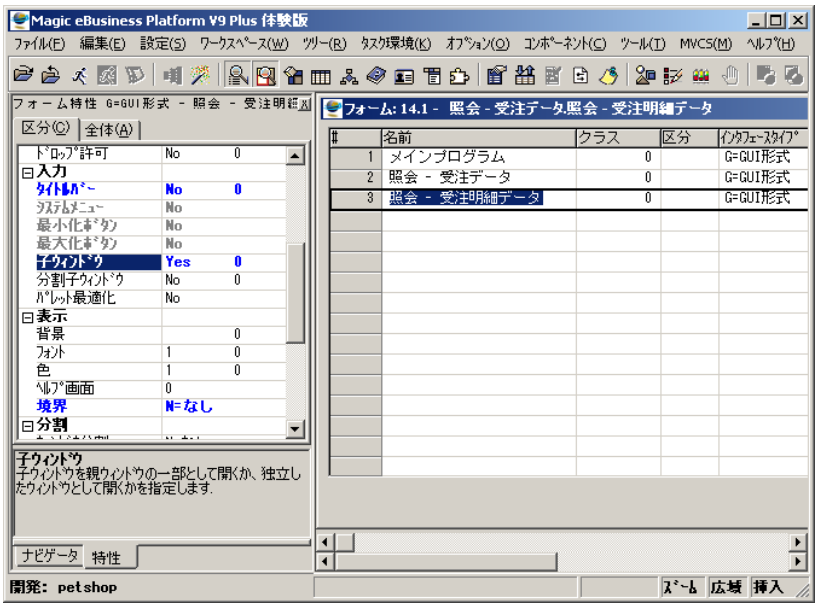


図 9-21 子ウィンドウ特性と、タイトルバー、境界特性の設定

参考： これらの特性は、フォームエディッタを開いた状態では変更することができません。上図のように、フォームエディッタを閉じて、フォームテーブルで変更してください。

9.2.7. 子タスクのフォームサイズと位置の調整

次に、フォームエディッタを開き、子タスクのフォームのサイズと位置を調整します。

子タスクにはタイトルバーをなくしたので、マウスのドラッグでウィンドウの位置を変更することができません。このため、フォーム特性の「上辺位置」「幅」「高さ」などの特性を直接数値で指定して、位置調整を行います。

最終的には、伝票らしく、下図のような感じにしてください。

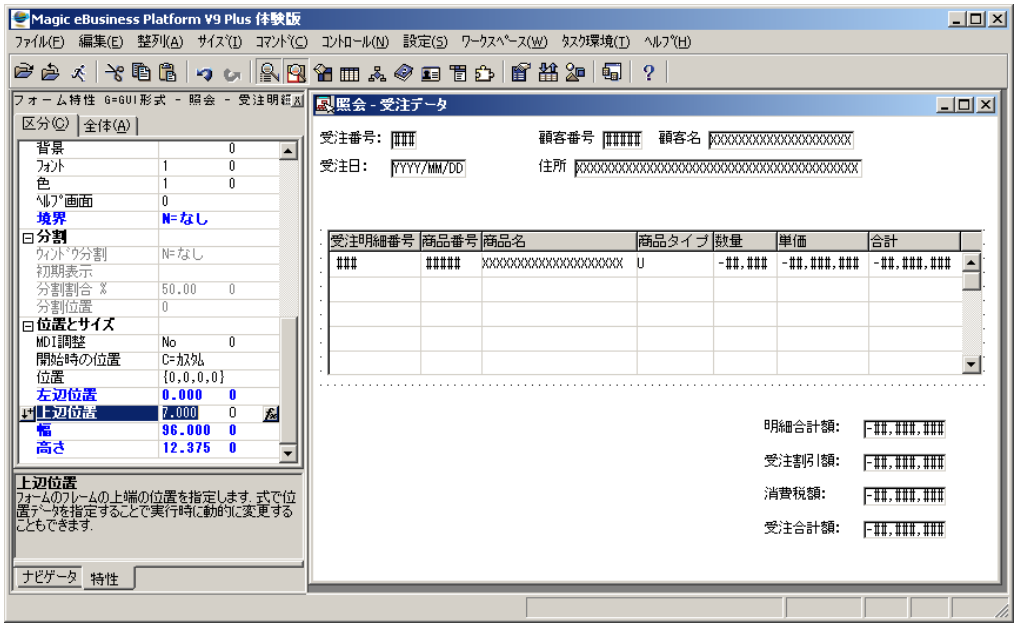


図 9-22 子タスクのフォームのサイズと位置の調整

9.3. 子タスクをいつ呼び出すか？

以上作ったプログラムを実行してみると、子タスクが呼び出されることがないので、親タスクのデータ(受注データ)だけしか表示されません。そこで、カーソルが顧客の住所から明細合計額に移るタイミングで呼び出されるようにしましょう。

前にも書いたように、フォーム上でのカーソルの動きは、レコードメインの「セレクト」コマンドと密接な関係があります。カーソルはセレクトコマンドで定義されている順に動くということは前に見ていきましたが、レコードメインの二つのセレクトコマンドの間に他のコマンド、例えばプログラム呼び出しを行う「コール」コマンドを入れると、カーソルが移動するタイミングでそのコマンドが実行されます。

今の例にあてはめてみれば、「住所」から「明細合計額」に移るタイミングで、子タスクの呼び出しを行いたいわけですので、「住所」を定義するセレクトコマンドと、「明細合計額」を定義するセレクトコマンドの間に、コールコマンドを入れてやればよいということになります。

ここでは、明細合計額を定義しているセレクトコマンドの直前に、コールコマンドを入れてみましょう。

1. 親タスクのレコードメインを開きます。
2. 「明細合計額」のセレクトコマンドの直前の行(受注日のセレクトコマンド)にカーソルを合わせます。
3. F4 で 1 行作成します。
4. 「C=コール」コマンドを選択します。(行の先頭でズームするとコンボボックスが出てくるので、そこから選びます。あるいは、単に C キーを押します)。
5. 「T=タスク」の欄は、コールするもののタイプを表します。今の場合、子タスクを呼び出すので、T=タスクのままで構いません。
6. 次の欄は、タスクの番号を指定します。ズームすると子タスクの一覧が表示されますので、その中から 1 番「照会 – 受注明細データ」を選びます。

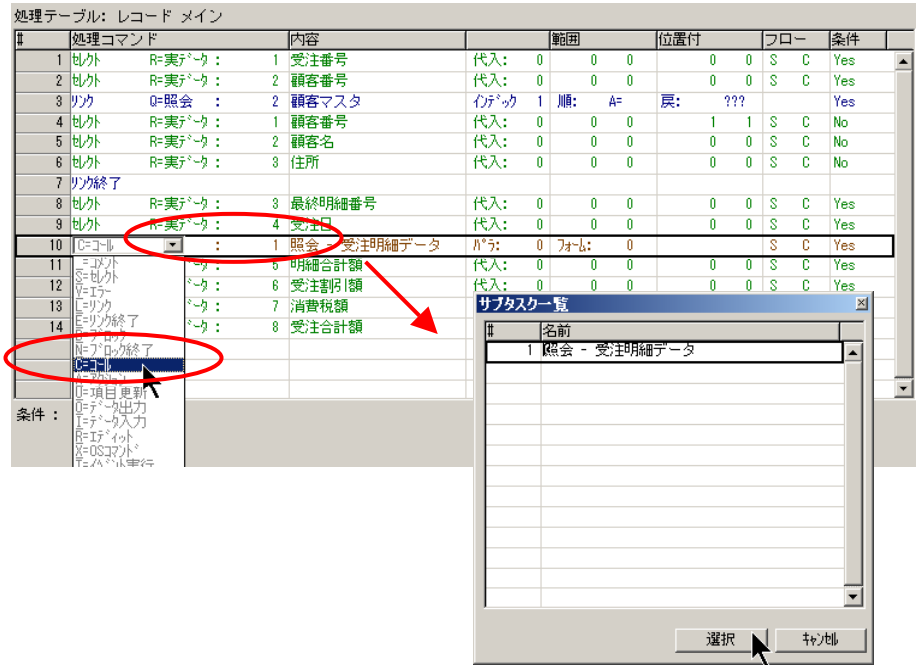


図 9-23 コールコマンドの追加

実行してみましょう。このプログラムを実行してみると、次のようになります。

1. 最初は、親タスクの画面だけが表示される。
2. Tab キーを何回か押すと、子タスクの画面が表示されるようになる。

3. Tab キーを押すと、子タスクの中でカーソルが移動する。
4. 親タスクに戻るには、ESC キーを押す。子タスクが終了し画面が消え、カーソルは親タスクの明細合計額に移ります。

9.4. 改善点

この動作では、まだ不便です。次のような改善が必要となります。

①. 子タスクが終了した後にも、明細行は残ったままにしたい。

②. プログラムを起動した状態から、明細行が表示されるようにしたい。

これを実現するため、Magic では次に説明するような機能を使います。

9.4.1. 子タスクの画面を残す

タスクが終了した後にも、そのときの状態をそのまま残像のように残しておく機能があります。これは、「**ウィンドウ消去**」パラメータを No に設定します。

1. プログラムを開きます。
2. ナビゲータで、子タスク「照会 – 受注明細データ」を選びます。
3. メニューで「タスク環境(K)」から「タスク制御(C)」を選びます（あるいは、Ctrl + C でもできます)。タスク制御ダイアログが開きます。
4. 「オプション(B)」タブをクリックします。
5. 「ウィンドウ消去：」パラメータを No にします。

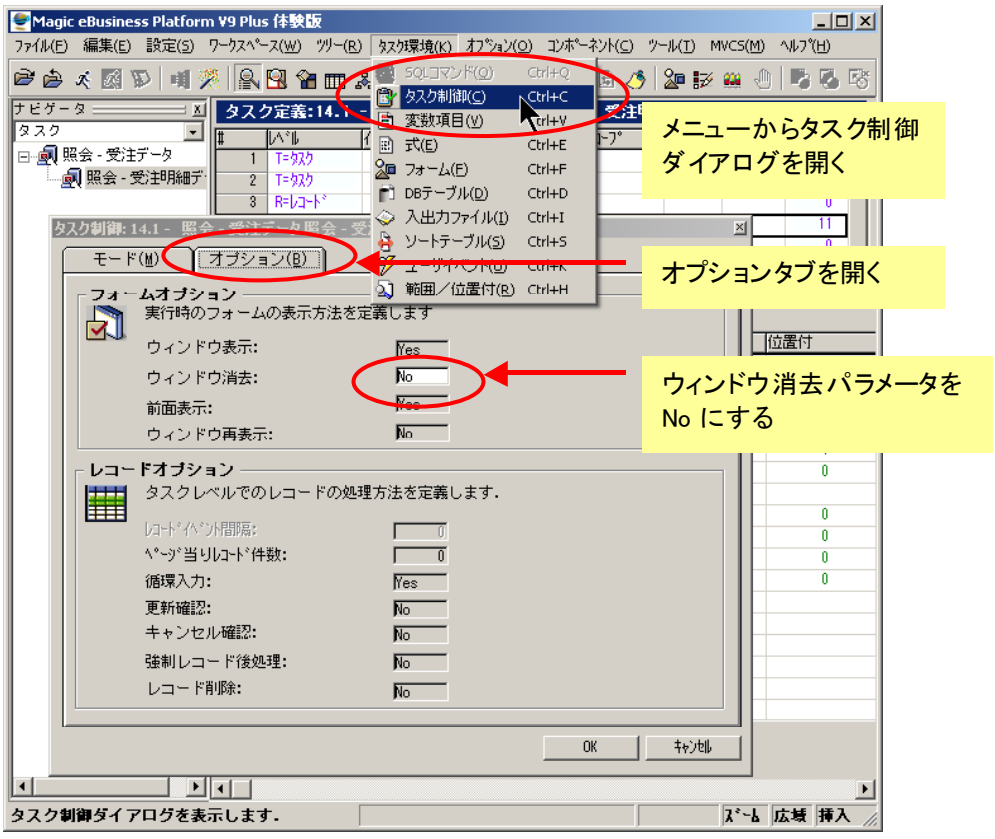


図 9-24 ウィンドウ消去パラメータの設定

これで、子タスクが終了した後にも、画面が閉じず、そのときのデータが残ることになります。実行して確認してみてください。

9.4.2. 始めから明細行が表示されるようにする

次に、プログラム開始直後から明細行が表示されるようにしましょう。
子タスクでは「ウィンドウ消去」を No に設定することにより、タスクが終了しても画面がそのまま残るようにしました。従って、最初から明細行を表示させるには、レコード処理の最初に子タスクを呼び出し、すぐに終了する、という処理を入れてやればよいことになります。これは次のようにします。

①. レコード前処理で、コールコマンドで子タスクを呼び出す。

レコード前処理というレベルは、初めて出てきますが、これは各レコードごとに、レコードがフェッチされた後、ユーザの入力が可能になる前に実行されるレベルです。レコードレベルでの初期化処理などを行う場合にこのレベルで定義します。
コールコマンドの定義の方法は、前にレコードメインで行ったのと全く同じです。

1. 親タスクのレコード前処理レベルを開きます。(「タスク定義」画面で、レベルが「R=レコード」、イベントが「P=前処理」の行をクリックします)。
2. 「処理テーブル： レコード前処理」画面をクリックします。カーソルがタイトル行に移ります。
3. F4 キーで新規行を作成します。
4. コールコマンドとし、タイプは「T=タスク」、タスク番号は 1 (照会 – 受注明細データ)とします。

図 9-25 レコード前処理で子タスクを呼び出し

②. 子タスクでは、すぐに終了する

レコード前処理で呼び出された子タスクはすぐに終了して、画面だけを残すことになります。ただし、子タスクはレコードメインからも呼び出され、このときにはすぐに終了しては困りますので、「親タスクのレコード前処理から呼び出された場合にはすぐに終了するが、そのほかの場合には終了しない」という条件をつける必要があります。
この条件は、子タスクの「タスク終了条件」で、「Level(1)='RP'」という式を指定することにより実現できますが、これについて少し詳しく説明します。

タスク終了条件

まず、タスク終了条件というパラメータについて説明します。

このパラメータは、タスク特性の「特性(P)」タブにあり、条件によってタスクを強制的に終了させることができます。デフォルトでは No となっていて、オンラインタスクの場合には、ユーザが ESC キーを押すか、あるいはウィンドウの右上にある X ボタンを押すなどをしない限りは終了しないようになっています。

このパラメータに式を指定した場合には、すぐ下にある「チェック時期」パラメータのタイミングで式が評価されます。デフォルトでは「B=前置」であり、レコード処理の最初(レコードフェッチの後、レコード前処理の前)に評価されます。

この式が False であれば、タスクはそのまま継続します。もし True となれば、タスクはその時点で終了します。

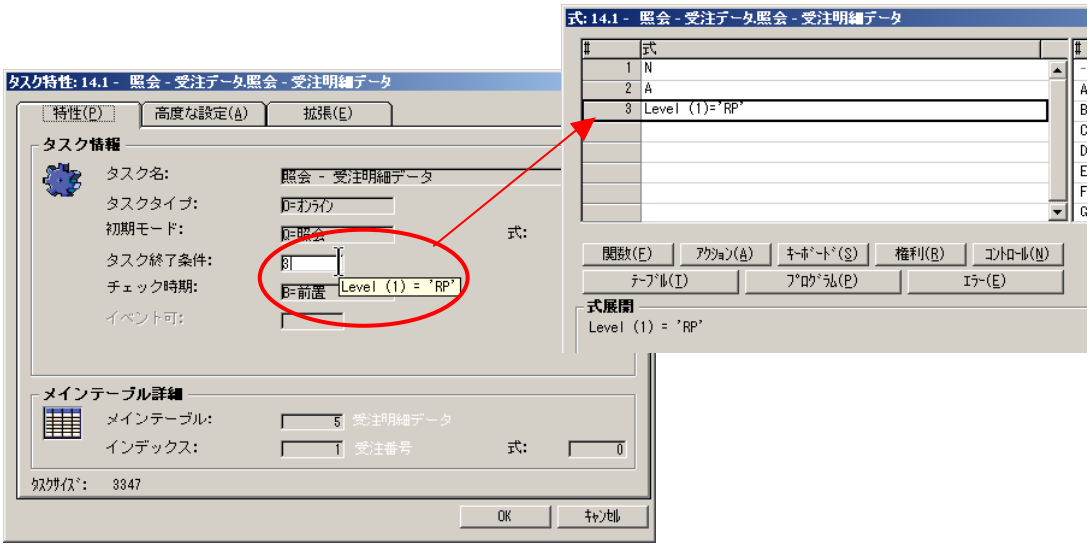


図 9-26 タスク終了条件

参考： 上図にあるように、「タスク終了条件」の欄にマウスカーソルを置くと、ツールチップが現れ、その式の展開形が表示されます。これはタスク終了条件に限らず、式を指定することのできる欄すべてに共通する、Magic の機能です。

Level() 関数

次に、Level() 関数についてです。Level() 関数は、タスクの実行レベルを文字列表現で返す関数です。パラメータとしては数値型の値をひとつとり、これはタスクの**世代**を示します。世代というのは、自タスクは 0、自分を呼び出しているタスクは 1、それを呼び出しているタスクは 2、・・・ という風に、実行時のタスクの呼び出し関係をさかのぼって増えていく番号のことです。

Level(1) と指定すると、自分を呼び出しているタスク、この例では、親タスクの実行レベルが文字列として返ります。返ってくる値は、主なものとして以下のような値があります。

戻り値	実行レベル
TP	タスク前処理
RP	レコード前処理
RM	レコードメイン
RS	レコード後処理
TS	タスク後処理

表 9-1 Level 関数の戻り値

従って、Level(1) ='RP' という式は、「親タスクの実行レベルがレコード前処理の場合には True、その他の場合には False」という結果になります。今の例では、この式をタスク終了条件に設定することにより、望みの動作を実現することができます。

以上の設定をしてから、プログラムを閉じて、実行してみましょう。今度は最初から明細行が表示されるようになります。

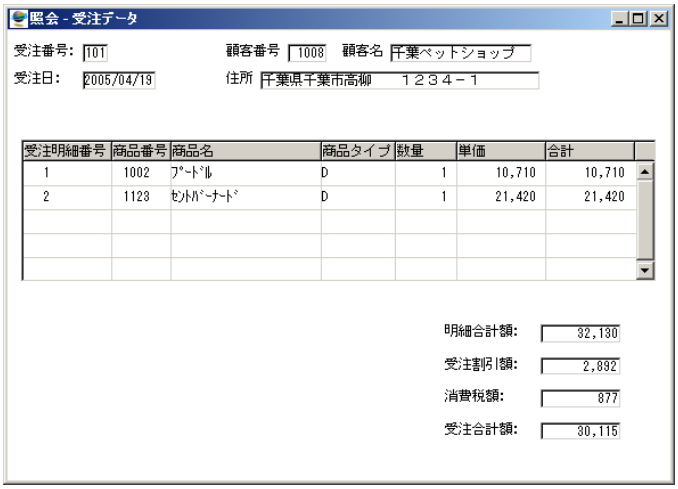


図 9-27 実行画面

PgUp および PgDn キーを使って、親タスクのレコードを移動してみてください。明細行の内容が、親レコードに連動して表示が変わることを確認してください。

また、Tab キーを何回か押していくと、子タスク(明細行)にカーソルが行くことを確認してください。

9.5. 最後の仕上げ

最後の仕上げとして、カーソルのパーク順序を調整しましょう。カーソルは表示の左上から右下への自然な順に従い、次の順序でパークするようにします。また、顧客名、住所など、マスターテーブルから取得してくるデータ項目にはパークしないようにします。

順序	項目	パークするかしないか？
1	受注番号	しない
2	受注日	する
3	顧客番号	する
4	顧客名	しない
5	住所	しない
6	明細合計額	しない
7	受注割引額	しない
8	消費税額	しない
9	受注合計額	しない

表 9-2 親タスクの項目のパーク順

順序	項目	パークするかしないか？
1	受注明細番号	しない
2	商品番号	する
3	商品タイプ	しない
4	数量	する
5	単価	しない
6	合計	しない

表 9-3 子タスクの項目のパーク順

前にも述べたように、フォーム上でのパークの順序は、レコードメインでのセレクトコマンドの順序と同じです。従って、親タスク、子タスクともに、上記の順序でセレクトコマンドを並べ替えることになります。

9.5.1. セレクトコマンドの順序の並べ替え方法

例として、親タスクのレコードメインの中で、「受注日」を定義するセレクトコマンドを、「受注番号」を定義するセレクトコマンドの直後に移動することを考えて見ます。

1. 親タスクのレコードメインを開きます。
2. 移動先の直前のコマンドにカーソルを移動します。今の場合、「受注番号」を定義しているセレクトコマンドにカーソルを置きます。

3. メニュー「編集(E)」から「移動登録(M)」を選びます。(Ctrl + M キーでも同じです)。「移動登録」ダイアログが表示されます。

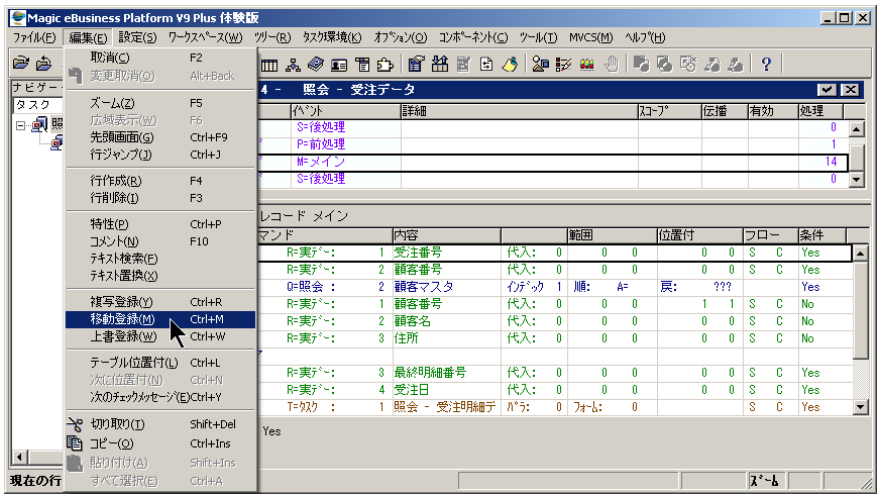


図 9-28 メニューから移動登録を選択

4. 「開始番号」欄からズームすると、レコードメインのコマンド一覧が表示されますので、移動したいコマンドにカーソルを合わせ、「選択」ボタンを押します。この場合、「受注日」のセレクトコマンドは 9 番目にありますので、9 が設定されます。

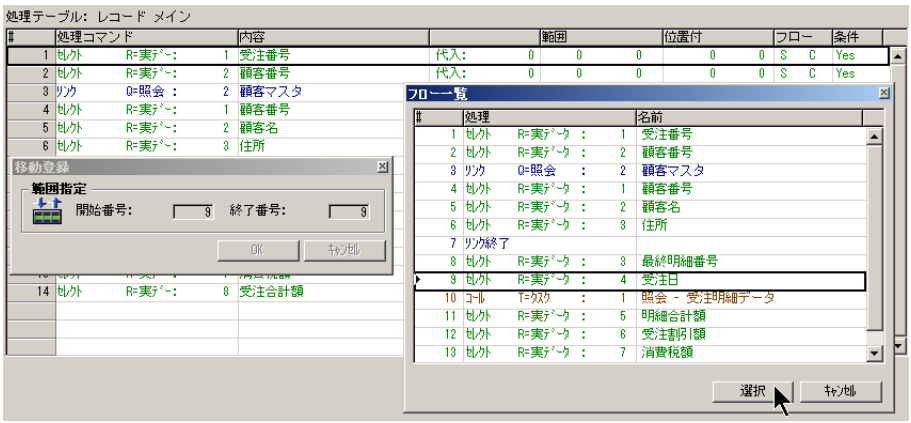


図 9-29 移動登録する番号を設定

5. 移動登録は、範囲を指定して複数の行の移動をいっぺんにすることができますが、今の例では 1 行だけ移動させるので、「終了番号」としては、開始番号と同じ 9 のままにしておきます。

6. OK を押すと、指定された行(受注日のセレクトコマンド)が、現在カーソルのある行の直後に移動されます。

図 9-30 セレクト「受注日」移動後のレコードメイン

同じ要領で、表 9-2 親タスクの項目のパーク順と表 9-3 子タスクの項目のパーク順に書いたように、セレクトコマンドを入れ替えてください。

9.5.2. パークする／しないの制御

フォーム上の項目にカーソルがパークするかしないかは、セレクトコマンドの「条件」欄の値に依存します。「条件」欄には Yes、No、あるいは式番号を設定することができます。条件が Yes または、実行時に True と評価される場合には、そこにカーソルがパークします。No あるいは False の場合には、カーソルはパークしません。

今回は、実行時にダイナミックにパークする／しないを変化させるような複雑な制御はしませんので、表 9-2 親タスクの項目のパーク順と表 9-3 子タスクの項目のパーク順 に従って、セレクトコマンドの「条件欄」に Yes あるいは No を設定してください。

参考： フォームに配置されていないデータ項目に対応するセレクトコマンドについては、「条件」欄がいずれに設定されてあっても動作には変わりがありません。

9.5.3. 実行してみる

これで参照用プログラムはできあがりです。実行してみて、設計したとおりの動作になっているかを確認してみてください。

10. 親子タスク２． 明細行の変更

前章で作成した親子タスクは、データの表示を行うだけのものでした。本章以降では、データの修正を行うことができるように、プログラムを修正していきます。

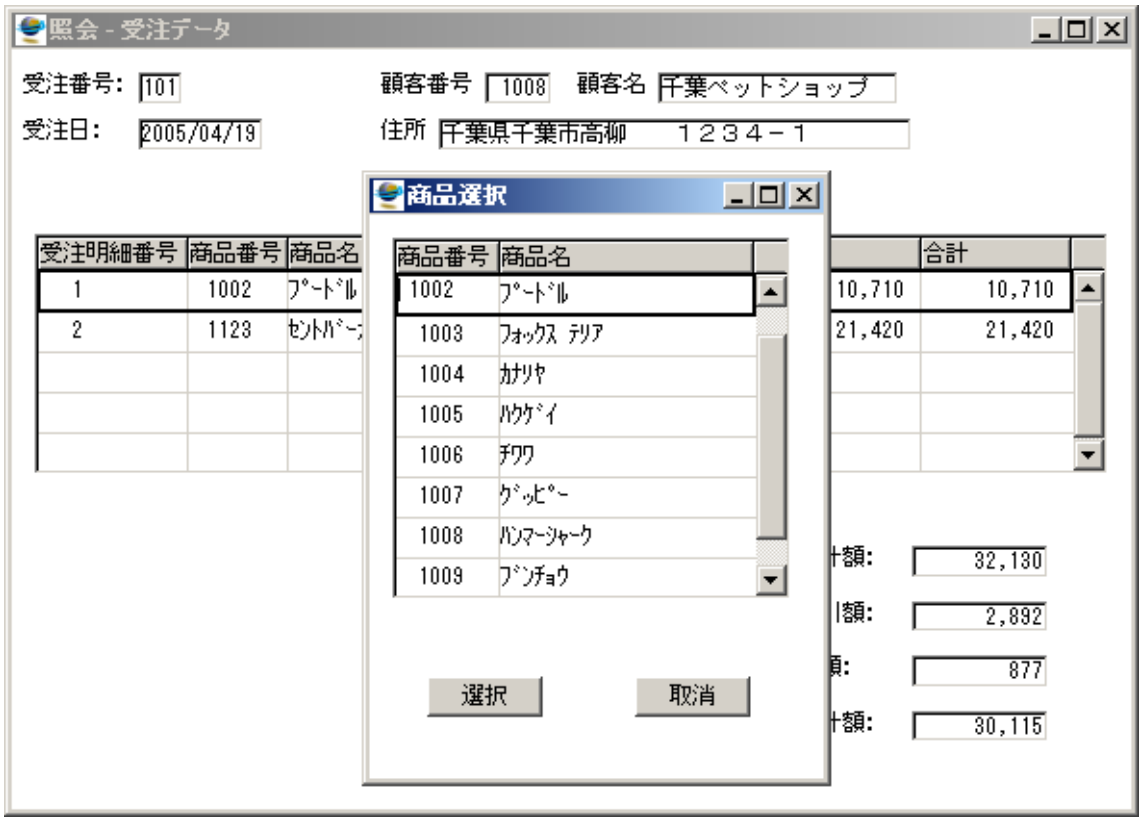


図 10-1 明細行データの変更

参考情報： 本章の内容については、リファレンスマニュアルにより詳細な情報がありますので参考にしてください。

再計算（代入式）	第 7 章 処理コマンド → セレクト → セレクトコマンドの設定欄 → 代入(オプション)
再計算（リンク）	第 7 章 処理コマンド → リンク → 概要 → リンク式の再計算
項目更新コマンド（加算モード）	第 7 章 処理コマンド → 項目更新 → 項目更新コマンドの設定欄 → 計： N=代入(デフォルト)、 および 使用の際の注意事項

10.1. 商品個数の修正

10.1.1. データ修正の波及

業務アプリケーションでは、ひとつのデータは別のデータと密接な関連を持っているのが普通で、データを修正するプログラムを作成する場合には、データの依存性を良く把握して、適切に再計算を行うことが必要です。最初に、一番簡単なケースとして、明細行の商品個数を変更する場合のことを考えてみます。例として、下図のようなデータを見てみます。

図 10-2 受注 #101 のデータ

ここで、明細行の最初の 1002 プードルの数量を 1 から 2 に変えたら、どういう波及が起こるでしょうか？

- 明細行の「合計」が、プードルの単価分(10,710 円)増える。
- 受注レコードの「明細合計額」が、同額増える。
- 受注割引額、消費税額、受注合計額が、それに連動して変わる。
- そのほかに、画面上では現れませんが、「顧客マスタ」には、「累計受注額」というカラムがあり、個々の顧客が今までに注文した額の合計が記録されていますので、上の受注データの「受注合計額」の変動に合わせ、1008 千葉ペットショップの「累計受注額」も更新する必要があります。
- もうひとつ、これも画面上では現れていませんが、「商品マスタ」には、各商品ごとに「受注数」を管理しています。従って、1002 プードルの受注数も ＋１ してやる必要があります。

このようなデータ変更の波及や、累計額の増減調整などは、ビジネスアプリケーションでは非常に多く現れる処理です。いわゆるビジネスロジックの多くの部分は、このようなデータ再計算・調整にあてられます。

10.1.2. データ項目の追加

Magic のプログラムでは、データの参照・修正を行う場合には、すべてレコードメインでセレクトコマンドを使ってデータ項目を定義しておく必要があります。前章までで作ったプログラムでは、データ再計算と更新のために必要となるデータ項目の定義が不足しているので、ロジックを書く前に、必要なデータ項目を追加しておきましょう。

追加が必要となる項目は、以下の通りです。

タスク	テーブル名	カラム名	備考
親タスク	顧客マスタ	割引率	受注割引額の計算のために必要。
		受注累計額	
	制御テーブル	消費税率	消費税額の再計算のために必要。
子タスク	商品マスタ	受注個数	

表 10-1 追加定義が必要なデータ項目

それぞれのタスクのレコードメインを開き、セレクトコマンドで上記のカラムを追加定義してください。

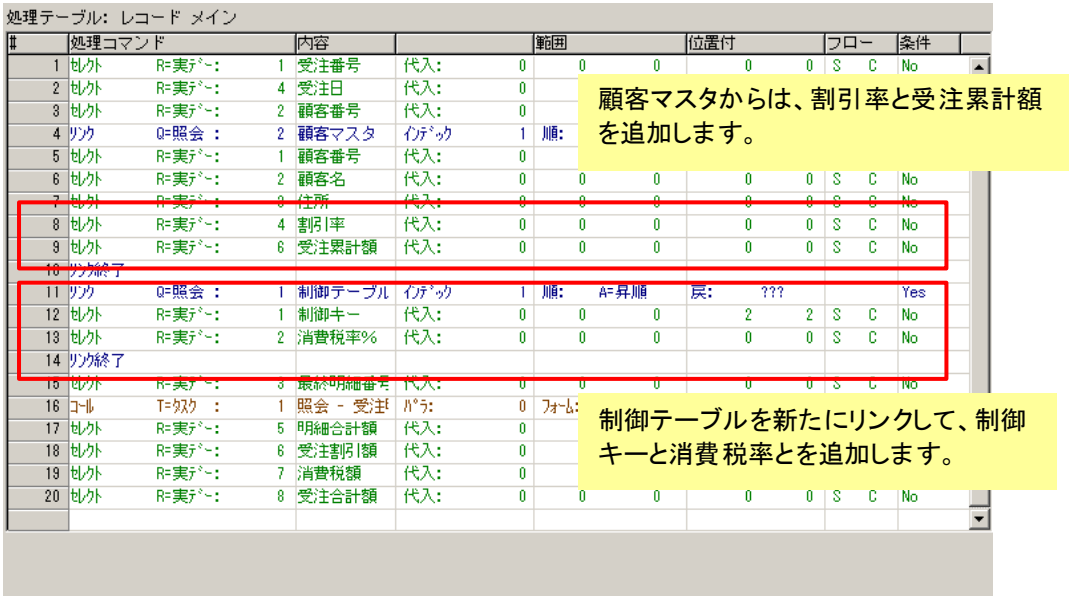


図 10-3 親タスクのレコードメインでの項目追加

図 10-4 子タスクのレコードメインでの項目追加

10.1.3. 明細の合計額の再計算

子タスクにおいて、明細行の「合計」の値は、合計 ＝ 単価 × 数量で計算され、常にこの関係が成立していなければなりません。このような恒等式の関係は、「代入式」欄で定義します。

定義のしかたは簡単です：

1. 子タスクのレコードメインを開きます。

2. 「合計」を定義しているセレクトコマンドの「代入」欄にカーソルを移動します。
3. ズームすると、式テーブルが開きます。
4. テーブル末尾に 1 行追加して、「W * X」とします。これは「数量 * 単価」です。
5. OK ボタンを押すと、式テーブルが閉じてセレクトコマンドに戻ります。「代入」欄には、今定義した式の番号 4 が設定されています。

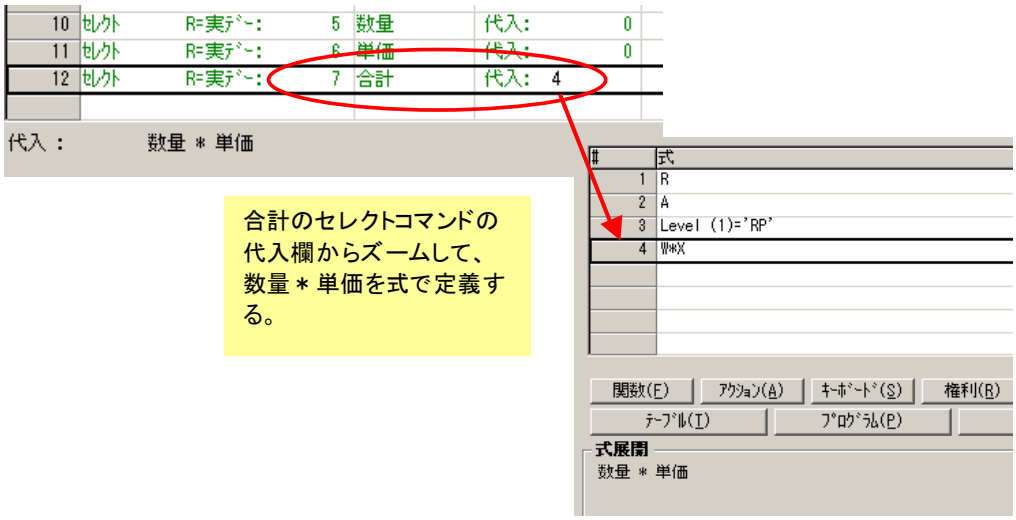


図 10-5 「合計」の代入欄に「数量 ＊ 単価」の式を設定

代入式で定義された式は、**再計算**の対象となります。再計算というのは、Magic の実行エンジンが自動的に行う機能の一つで、式中の変数の値がなんらかの原因で変わったときに、式が自動的に計算され、「代入」で設定されているデータ項目に代入されます。
今の例で言えば、「数量」の値が、ユーザの入力により 1 から 2 に変わったとしますと、「数量」を使っている式 4 が自動再計算され、「合計」の項目に代入されます。従って、「合計」額は常に正しい値に維持されるようになります。

10.1.4. 商品マスタの受注数の更新

次に、「数量」が更新されたならば、商品マスタの「受注数」も更新しなくてはなりません。
今回の場合は、上に説明した代入式ではうまく計算されません。なぜならば、「受注数」というのは、今までの累計個数であるので、受注明細の「数値」が 1 増えたならば、商品マスタの「受注数」も 1 増やす、というように、初めの値と終わりの値との**差分**を増減しなければならないからです。
幸いなことに Magic には、このような差分の増減調整のためにも大変便利な機能が用意されています。それが「加算」モードの項目更新コマンドです。

加算モードの項目更新コマンド

項目更新コマンドは以前にも出てきましたが、普通のプログラミング言語の代入文と同じで、式を計算し、結果を変数(Magic の場合にはデータ項目)に代入するものです。
通常のモードの項目更新コマンドでは、単純に式を計算しデータ項目に代入するだけなのですが、「加算」モードの場合には、レコードの初期状態を憶えていて、その初期状態で計算した式の値と、項目更新コマンドを実行した時点で計算した式の値との両方を計算し、その差分を、指定されたデータ項目に加える、という動作になります。

具体的に、例で見てみましょう。レコードの初期状態では、次のようになっていました。(関係のない部分は省略しています)

- 商品明細レコード (初期値)

受注番号	受注明細番号	商品番号	数量	単価	合計
101	1	1002	1	10,710	10,710

- 商品マスタ (初期値)

商品番号	商品名	・・・	受注数	・・・
1002	プードル	・・・	1	・・・

ここで、ユーザが数量を 1 から 2 に変更したとします。代入式により、合計額が再計算され、更新されます。

- 商品明細レコード (変更後)

受注番号	受注明細番号	商品番号	数量	単価	合計
101	1	1002	2	10,710	21,420

この状態で、次のような「加算」モードの項目更新コマンドを実行します。

項目更新 U (受注数)　式： 数量、　計算モード： 加算

そうすると、「式」は単に「数量」ですので、数量の初期値と変更後の値とから、差分を計算します。

（差分値）＝（変更後の値）― （初期値）＝ 2 － 1 = 1

従って、U (商品マスタの受注数)には、1 が加算され、結果は 2 となります。

- 商品マスタ (加算モードの項目更新後)

商品番号	商品名	・・・	受注数	・・・
1002	プードル	・・・	2	・・・

このようにして、正しく受注数が更新されます。

子タスクでの利用

この項目更新コマンドは、レコード後処理のレベルで実行されるのが普通です。このレベルは、ユーザの入力がレコードメインで終わり次のレコードに移ろうとするとき、データベースへのレコードの更新を行う前に実行されるレベルで、主にレコード単位での、文字通り「後処理」を行う場所です。

1. 子タスクの「R=レコード」「S=後処理」を開きます。
2. 処理テーブルに、 F4 で 1 行追加します。
3. 行頭でズームしてコンボボックスから「U=項目更新」を選びます(あるいは、単に U と入力しても同じです)。
4. 次の欄は更新項目の欄で、ズームするとデータ項目一覧が表示されるので、商品レコードの受注数を選択します。
5. 「式：」の欄からズームすると、式テーブルが開きます。
6. テーブル末尾に 1 行追加し、W (数量) と定義します。
7. OK ボタンを押すと、項目更新コマンドに戻り、今定義した式番号 5 が設定されます。
8. 次の「計：」の欄が計算モードの欄で、デフォルトは「N=代入」ですが、加算更新モードにするために、「I=加算」を選びます。

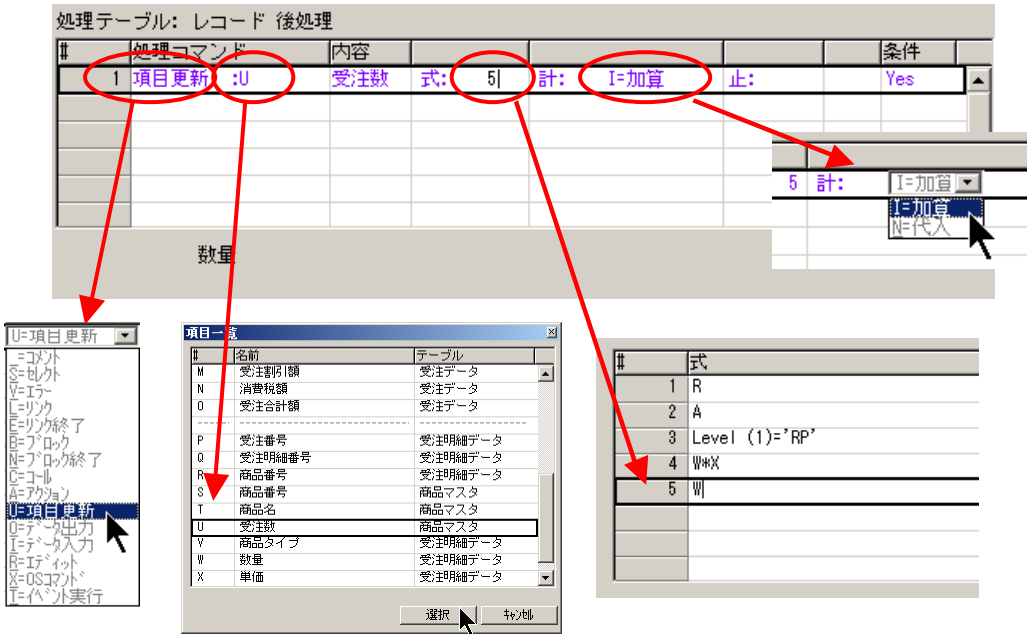


図 10-6 子タスクのレコード後処理での、加算モード項目更新（商品マスタの更新）

10.1.5. 受注レコードの明細合計額の更新

明細行についてはこれで終わりですが、明細行の「合計」額が変動することにより、親タスクの受注レコードの「明細合計額」を更新する必要があります。この値は、常に、明細行の「合計」の値の合計を格納しているからです。

この値を更新するのも、「加算」モードの項目更新コマンドを使えば、たやすく定義することができます。すなわち、「合計」額が変動したときには、その差分を「明細合計額」に加算するわけですから、加算の項目更新コマンドを使って、

　　項目更新　L(明細合計額)、式： (合計)、計： I=加算

というものを、子タスクのレコード後処理に入れることになります。

最終的に、子タスクのレコード後処理は、次のようになります。ここで式 6 は、Y(受注明細の「合計」カラム)です。

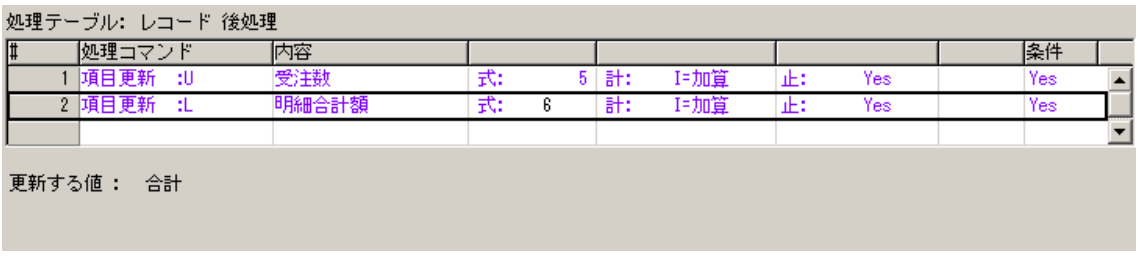


図 10-7 子タスクのレコード後処理

10.1.6. 親タスクでの再計算

子タスクで、明細合計額を更新しましたが、これに伴って、親タスクではいくつかの再計算を行う必要があります。

受注割引額 ＝ 明細合計額 ＊ 割引率 / 100

消費税額 ＝ （明細合計額 － 受注割引額）＊ 消費税率 / 100

受注合計額 ＝ 明細合計額 － 受注割引額 ＋ 消費税額

これらは常に成り立っていなければならない恒等式ですので、セレクトコマンドの「代入」式に定義しておけば、Magic が自動で再計算してくれることになります。

最終的に、親タスクのレコードメインは次のようになります。

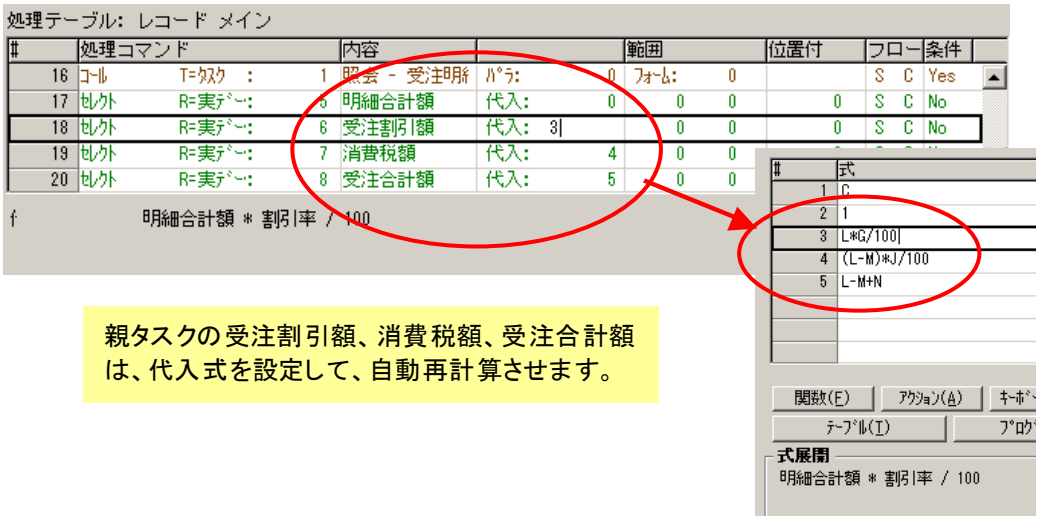


図 10-8 親タスクのレコードメインでの、代入式設定

10.1.7. 顧客マスタの更新

最後に残ったのは、顧客マスタの「受注累計額」です。この値には、ある顧客から受注した額の合計を格納しておきます。従って、ある注文の「受注合計額」が変更された場合には、それに従って受注累計額も更新する必要があります。

今までも累計値の維持管理には、加算モードの項目更新コマンドを使いましたが、受注累計額でもこれを使いましょう。親タスクのレコード後処理で、

項目更新　H (受注累計額)、式： (受注合計額)、計： I=加算

と定義すればできあがりです。

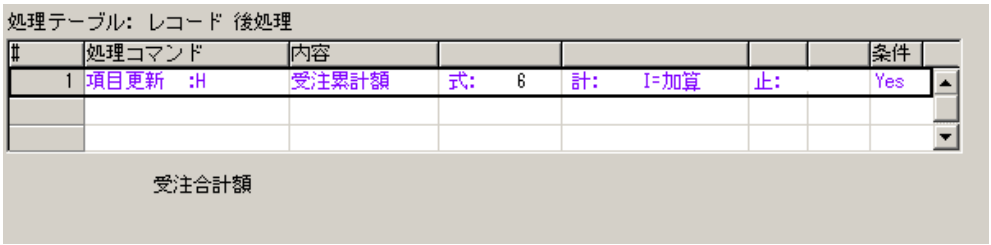


図 10-9 親タスクのレコード後処理での受注累計額更新

10.1.8. タスクモードの変更

最後ですが、一番基本的な変更を忘れずにしておきましょう。タスクの初期モードです。

タスク特性を開いて「初期モード」を見てみると、親タスク、子タスクともに「Q=照会」となっています。照会モードは、データの読み込み専用で修正はできないモードですので、修正ができるようにしましょう。

- 親タスクは、「M=修正」にします。
- 子タスクは、「P=親と同じ」にします。

子タスクの「P=親と同じ」というモードは、文字通り、親タスクのモードと同じにする、ということで、親が照会モードであれば子も照会モードとなり、親が修正モードであれば子も修正モードとなります。

本章で作ってきた 1：N 関係の親子タスクでは、親と子とが同じモードで動くというのが普通ですので、後の章で登録モードになることも考慮して、「P=親と同じ」としておくのが便利です。

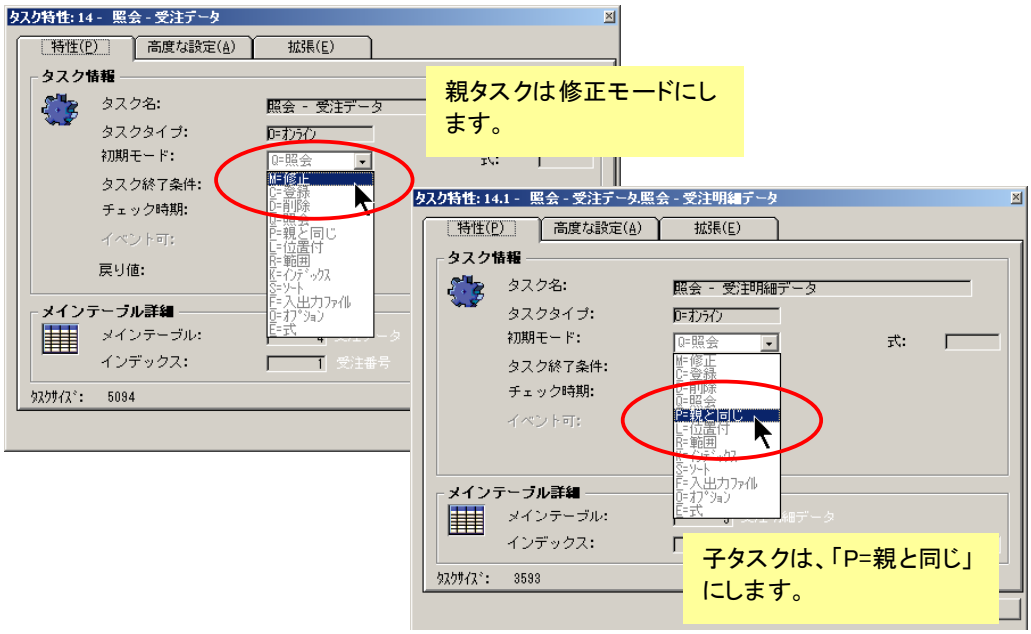


図 10-10 タスク初期モードの変更

10.1.9. 実行してみる

F8 キーでシンタックスチェックを行い、大丈夫ならば、実行して動作を確認しましょう。

実行する前に、本章で追加したロジックが正しく動作しているのかを確認するために、変更前のデータの値を記録しておきましょう。テーブルリポジトリで APG を行い、データを見てみると、以下のようになっています。(関係ない部分は省略しています)

- 顧客マスタ（初期値）

商品番号	商品名	割引率	受注累計額
1008	千葉ペットショップ	9.00	30,115

- 商品マスタ（初期値）

商品番号	商品名	受注数
1002	プードル	1

- 制御データ（初期値）

キー	消費税率
1	3.00

この状態で

1. 受注入力プログラムを起動します。
2. 明細行の第 1 行目(1002 プードル)に入ります。

3. 「数量」を 1 から 2 に変更します。→ 「合計」が 10,710 から 21,420 になります。
4. ESC キーを押して、親タスクに戻ります。→ 明細合計額から受注合計額までが更新されます。
5. ESC キーをもう一度押して、プログラムを終了します。

図 10-11 数量変更後の画面

実行時に、画面上で次の数値を確認してください。

- 明細合計額は、明細行の「合計」の合計で、21,420+21,420=42,840
- 受注割引額は、この顧客の割引率が 9%なので、42,840 * 9/100 = 3,856
- 消費税額は、消費税率が 3%なので(古いですね)
  （明細合計額 － 受注割引額）* 3 / 100 ＝ 1,170
- よって、受注合計額は、明細合計額 － 受注割引額 ＋ 消費税額 = 40,154

となっているはずです。

また、実行後に、顧客マスタと商品マスタを APG で見てみて、次の数字を確認してください。

- 顧客マスタでは、1008 千葉ペットショップの受注累計額が 40,154
- 商品マスタでは、1002 プードルの受注数が 2

となっていれば、正解です。

以上、明細行の「個数」を修正できるようにするために、プログラムを修正してきました。たかが、一項目の修正をできるようにするためだけに、ずいぶんと大変な修正が必要なんだなあ・・・、これからどうなるんだろう？とお思いかと思います。

でも、実のところは、これができるようになるだけで、ロジックのかなりの部分が出来上がってしまっているのです。次にそれを見ていきましょう。

10.2. 商品の変更

今までは、明細行の「数量」の変更だけを考えてきました。次には、商品番号も変更できるようにしたいところです。

10.2.1. 商品番号変更時のデータ変更の波及

例として、商品番号と数量とを同時に変更したときに、どんな波及があるかを考えて見ましょう。図 10-12 を見てください。これは、受注 101 の最初の明細行で、商品番号を 1002 から 1003 に変更し、数量を 2 から 1 に変更する場合です。

① 商品番号が 1003 に変更されるので、商品マスタからは 1003 の商品を読み込んでくる。
② 商品番号 1003 のレコードから、商品タイプ、単価を読み込み、受注明細データのレコードにコピーする。
③ 受注明細データのレコードでは、数量と単価から、合計を再計算する。
④ 受注明細データの合計の変更に伴い、受注データの明細合計を更新する。
⑤ 受注データでは、明細合計の変更に伴い、受注割引額、消費税額、受注合計額を更新する。
⑥ 商品番号 1002 のレコードの受注数は 2 → 0 とし、商品番号 1003 のレコードの受注数は 0 → 1 とする。

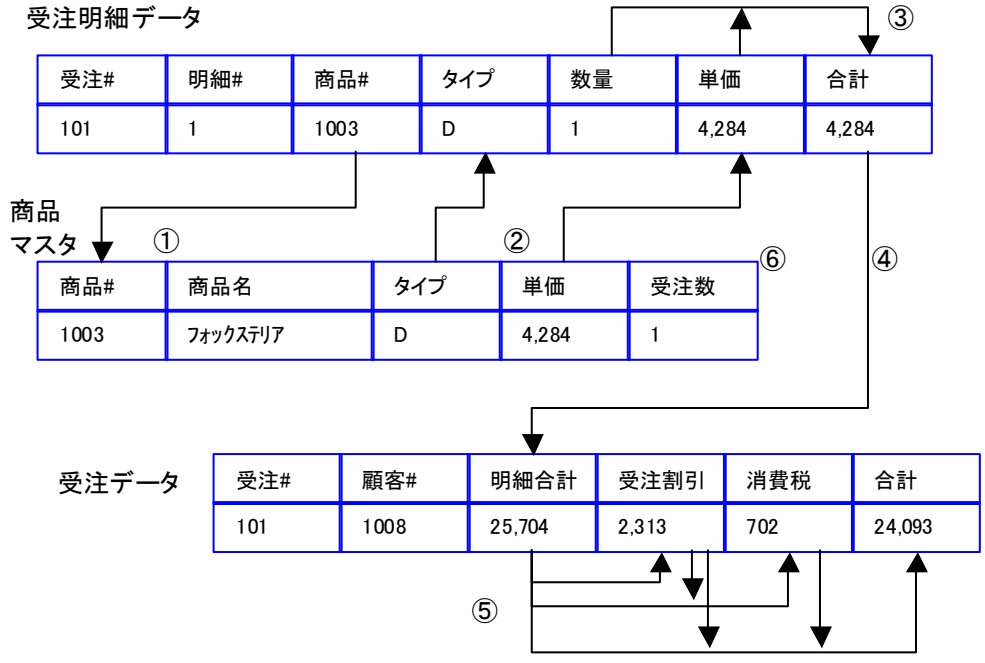


図 10-12 商品番号変更に伴うデータ更新の波及

10.2.2. データ変更波及への対応

では、これに対応して、どのようにプログラムを修正するかを見ていきます。

① 商品番号が 1003 に変更されると、商品マスタから 1003 の商品を読み込んでくる。

これに対しては、実は何もプログラムを変更する必要はありません。Magic のデータリンクでは、リンク条件

(今の場合、商品番号が等しい、という条件)のデータに変更があった場合、自動的に**リンクの再計算**を行います。このため、商品番号が 1002 から 1003 に変わった場合には、Magic エンジンが自動的にリンク再計算を行い、1003 のレコードを読み込んできます。

② 商品番号 1003 のレコードから、商品タイプ、単価を読み込み、受注明細データのレコードにコピーする。

これは、以下のような恒等式の関係にあるので、代入式を使います。

- 受注明細データの「商品タイプ」＝商品マスタの「商品タイプ」
- 受注明細データの「単価」＝商品マスタの「単価」

次のようにして、子タスクのレコードメインで、商品タイプと単価の「代入:」式に設定してください(図 10-13 参照)。

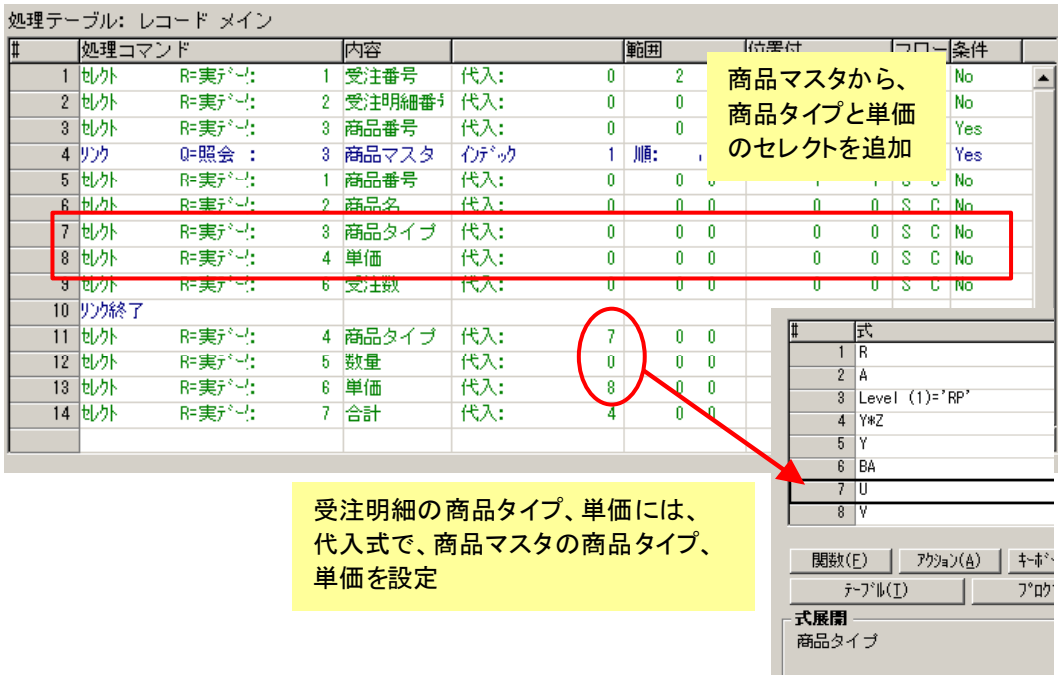


図 10-13 商品タイプと単価を商品マスタからコピー

1. 「リンク Q=照会 3 商品マスタ」の下、「セレクト 商品名」の後に、二つセレクトコマンドを追加し、3 (商品タイプ)と 4 (単価) をセレクトします。
2. 「リンク終了」の後の「セレクト 商品タイプ」の「代入:」カラムには、U (商品マスタの商品タイプカラム)の式を指定します。
3. 「セレクト 単価」の「代入:」カラムには、V (商品マスタの単価カラム)の式を指定します。

このように設定することにより、商品番号が変更されたときには、データリンクの再計算が起こり、それに伴い、商品タイプと単価の代入式が再計算されるので、受注明細レコードにも商品タイプと単価がコピーされることになります。

③ 受注明細データのレコードでは、数量と単価から、合計を再計算する。

これについては、何もする必要はありません。「合計」項目には、すでに代入式として、(単価) × (数量) という式が設定されており、単価が変わった場合には、数量が変わった場合と同様、自動的に Magic エンジンが再計算を行うからです。

④ **受注明細データの合計の変更に伴い、受注データの明細合計を更新する。**

これについてもプログラム変更は不要です。受注データの明細合計に明細行の「合計」の差分を加算するというロジックは、すでにレコード後処理の「加算」モードの項目更新により実現されているからです。

⑤ **受注データでは、明細合計の変更に伴い、受注割引額、消費税額、受注合計額を更新する。**

これも、プログラム修正は不要です。このロジックもすでに実装されているからです。

⑥ **商品番号 1002 のレコードの受注数は 2 → 0 とし、商品番号 1003 のレコードの受注数は 0 → 1 とする。**

1002 のプードルは注文が取り消されたので、受注数は－2しなければなりません。一方、1003 のフォックステリアは新たに1匹注文されたので、受注数を＋1する必要があります。

これはどう Magic で記述するのでしょうか？ちょっと複雑そうですね。

実は、これも何もする必要がありません。子タスクのレコード後処理では、加算モードで「受注数」の更新を行っていますが、加算モードの項目更新コマンドでは、リンクされたレコードの項目を更新する場合、リンクが初期状態から変わったかどうかまで、チェックしています。

一般則として、今の場合のように商品番号が変わったことによりリンクの再計算が起こり、別レコードがリンクされた場合には、

- 初期状態のレコードについては、0 になったものとして差分を取る。
- 再計算後のレコードについては、初期状態は 0 だったものとして差分を取る。

という計算をして、それぞれのレコードに対して差分の加算を行ってくれます（図 10-14)。

商品マスタレコード	初期状態	終了状態	差分
#1002	2	リンクから外れたので、0とみなす	－2
#1003	初期状態でリンクされていなかったので、0 とみなす。	1	＋1

図 10-14 リンクの再計算が起こった場合の、加算モードの項目更新

10.2.3. 実行してみる

実際にやってみましょう。受注番号 101 の、明細行#1 は 1002 プードルで、先の修正により、数量は 2 になっていますが、これを、商品番号・数量ともに変更してみます。

1. プログラムを実行します。最初の受注番号 101 が表示されます。
2. 明細行#1 に移動します。商品番号 1002 プードル、数量は 2 です。
3. 商品番号の項目でズーム (F5 またはダブルクリック)してみてください。商品一覧が開きます。(ちなみに、プログラムでは何もしていないのに、何故選択プログラムが開くのでしょうか？？ モデルリポジトリで、商品番号モデルに選択プログラムを定義してあるからです)。
4. 1003 フォックステリアを選びます。ここで、単価と合計が自動再計算されることを確認してください。
5. 数量の項目を 1 にします。ここで、合計が再度再計算されることを確認してください。
6. ESC で親タスクに戻ります。ここで、親タスクの「明細合計額」から「受注合計額」がすべて再

計算されることを確認してください。

7. 再度 ESC を押して、プログラムを終了します。
8. テーブルリポジトリを開きます。
9. 商品マスタで APG を行い、1002 プードルの受注数は 0、1003 フォックステリアの受注数は 1 になっていることを確認してください。

10.3. 明細行の追加

次には、明細行を新規に追加できるようにしましょう。
今まで作ってきたプログラムで、かなりの部分がすでにできあがっていますが、新規追加を行う場合に抜けている処理は、受注番号と受注明細番号の設定です。
受注明細データは、受注番号と受注明細番号とを一意(重複不可)のキーとして定義されています。新規にレコードを作成した場合には、受注番号と受注明細番号とを、テーブル内で一意になるように設定してやらなければなりません。

10.3.1. 受注番号の設定

最初に、受注番号です。明細行の受注番号のカラムは、親の受注レコードの受注番号と常に等しいものですので、代入式で親の受注番号 (A)を設定してやります。
新規レコード作成時、すなわち**登録モード**の時には、代入式は常に計算され、そのカラムの初期値として設定されます。ちなみに、代入式が設定されていない項目は、登録モードでは、空文字あるいは数値の 0 が初期値として設定されます。
子タスクのレコードメインを開き、先頭の「セレクト 1 受注番号」の「代入」式に、親レコードの受注番号を式で設定してやるのですが、式テーブルを見てみると、式の 2 番に、すでに A は作成されています。この式は、同じセレクトコマンドの「範囲」開始・終了の指定に使われていますが、これをそのまま流用してしまいましょう。すると、このセレクトコマンドは、代入式、範囲開始・終了欄に、すべて式 2 が設定されることになります（図 10-15 参照)。このような作りは、親子タスクの子タスクでは非常によく出てくるパターンです。

図 10-15 明細行の受注番号の代入式で、親タスクの受注番号を設定

10.3.2. 明細番号の発番

次には、明細番号の発番です。すなわち、各受注データ内で、明細番号を連番で自動的に発番する機能です。このプログラムでは、次のようにして明細番号を発番しています。

- 受注データのレコードには、「最終明細番号」というカラムがあり、この受注での最終明細番号を記憶しています。
  このカラムは、親タスクですでにセレクトコマンドで選択されているので、改めて定義する必要はありま

せん。

- 子タスクが登録モードの場合には、**レコード前処理**で「最終明細番号 ＋ 1」を受注明細番号として設定します。これは、通常の項目更新コマンドを使います。ただし、「登録モードの場合」だけ、という条件付けを行う必要があります。この条件は、項目更新コマンドの「条件」欄に、「Stat (0, 'C'MODE)」という式を使って指定します。

  参考： Stat(世代、モード) という関数は、「世代」で指定されたタスク(自タスクは 0、自タスクを呼び出したタスクは 1、それを呼び出したタスクは 2、・・・)が、「モード」で指定されたモードであれば True、そうでなければ False を返します。モードとして指定できるのは、'C'MODE (登録モード)、'M'MODE (修正モード)、'Q'MODE' (照会モード)があります。

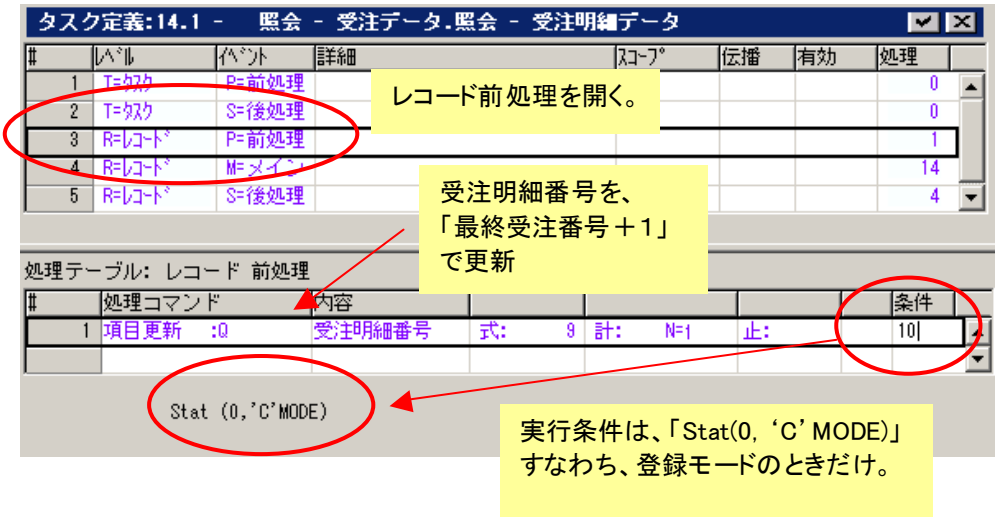


図 10-16 登録モードでは、レコード前処理で受注明細番号を設定

- 最後に親レコードの最終明細番号を更新します。
  これは、レコード後処理で、項目更新コマンドで行います。ここでも「登録モードの場合にだけ」という条件付けが必要になりますが、先ほどと同じく、Stat 関数を使って設定します。

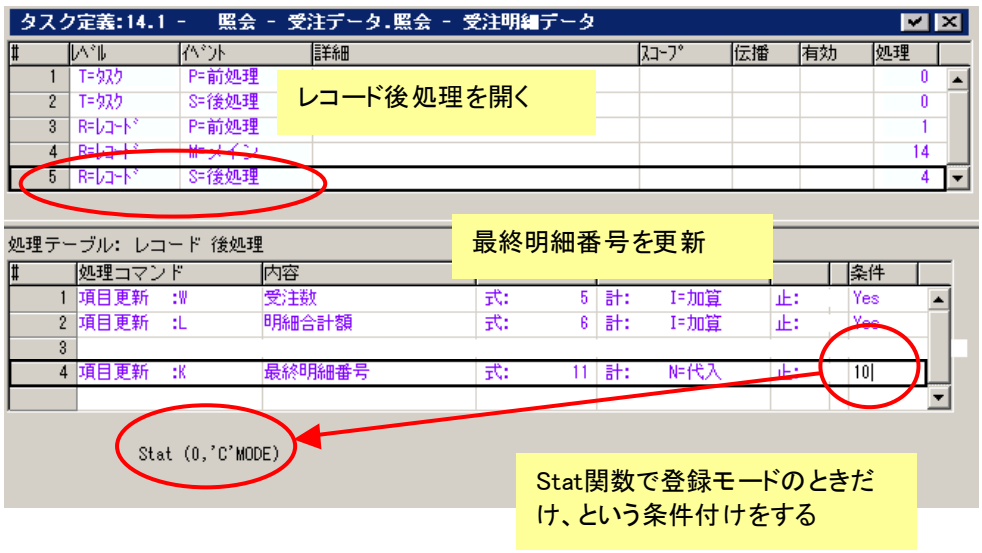


図 10-17 最終明細番号は、子タスクのレコード後処理で更新

10.3.3. 登録時の加算の項目更新について

図 10-17 にあるように、子タスクのレコード後処理では、商品マスタの受注数と親の受注レコードの明細合計額に対し、加算モードで項目更新を行っています。
この二つの項目更新コマンドは、登録モードでも予期したように動作するのでしょうか？
登録モードの時の、加算モードの項目更新では、初期値は 0 として扱います。従って、差分としては現在値そのものとなりますから、例えば商品 1004 カナリアを 1 羽追加注文した場合には、1004 のレコードの受注数は、＋1 になります。これは正しい動作です。
同じように、親レコードの明細合計額への加算の項目更新も、正しい動作であることがわかります。
このように、加算の項目更新コマンドは、登録モードでも問題なく動作することがわかります。

10.3.4. 実行してみる

実際に、プログラムを起動し、注文 101 に、1004 カナリアを 1 羽追加して、すべての計算が正しく行われることを確認してください。

10.4. 明細行の削除

本章の最後として、明細行を削除できるようにしてみましょう。

10.4.1. 明細行削除時のデータ変更波及

明細行を削除すると、次のようなデータ更新の波及が起こります。

① 商品マスタの「受注数」を、削除前の「数量」分だけ減算する。

② 親の受注データレコードの「明細合計額」を、明細行の「合計」分だけ減算する。

③ 明細合計額の変更に伴い、受注割引額、消費税額、受注合計額も再計算する。

注意: ここでは､最終受注番号の調整とか､あるいは途中の行を削除した場合の連番つけなおしなどは、簡単のために行わないことにします。

これらのデータ更新の波及への対応としては、実は何もする必要がありません。今まで作成してきたロジックで、すべて吸収されています。①と②は、子タスクのレコード後処理での二つの加算モード項目更新コマンドにより、③は親タスクでの代入式の設定により、すでにロジックが定義されているからです。

10.4.2. 削除時の加算の項目更新

ひとつだけ説明事項があります。それは、削除時の加算の項目更新です。子タスクでは、レコード後処理で加算モードの項目更新コマンドがありますが、削除時にこれはどう働くのでしょうか?

登録モードでは、初期値が 0 として計算されるので、差分は現在値そのものでした。削除モードでは反対に、現在値が 0 として計算されます。削除されるので、0 になるのと同じだからです。従って、差分は 0 －(現在値)＝ －(現在値)となります。

例えば、先に受注 101 に追加した、カナリア 1 羽を削除してみましょう。レコード後処理で、商品マスタの受注数に対する加算の項目更新コマンドが実行されるとき、現在値は 1 なので、差分は上記のように －1 となり、商品マスタのカナリアの受注数が －1 されます。注文がキャンセルされたのですから、これは正しい動作です。

このように、加算モードの項目更新コマンドは、修正、登録、削除すべてにおいて、さらにはリンクが再計算された場合においても、すべて正しく動作するように設計されている、きわめて賢いオペレーションです。

11. 親子タスク３． 注文票の変更

前章までで、受注明細行の追加・修正・削除にまで対応しました。次には、親タスクの受注レコードを修正・追加・削除できるようにプログラムを修正していきましょう。

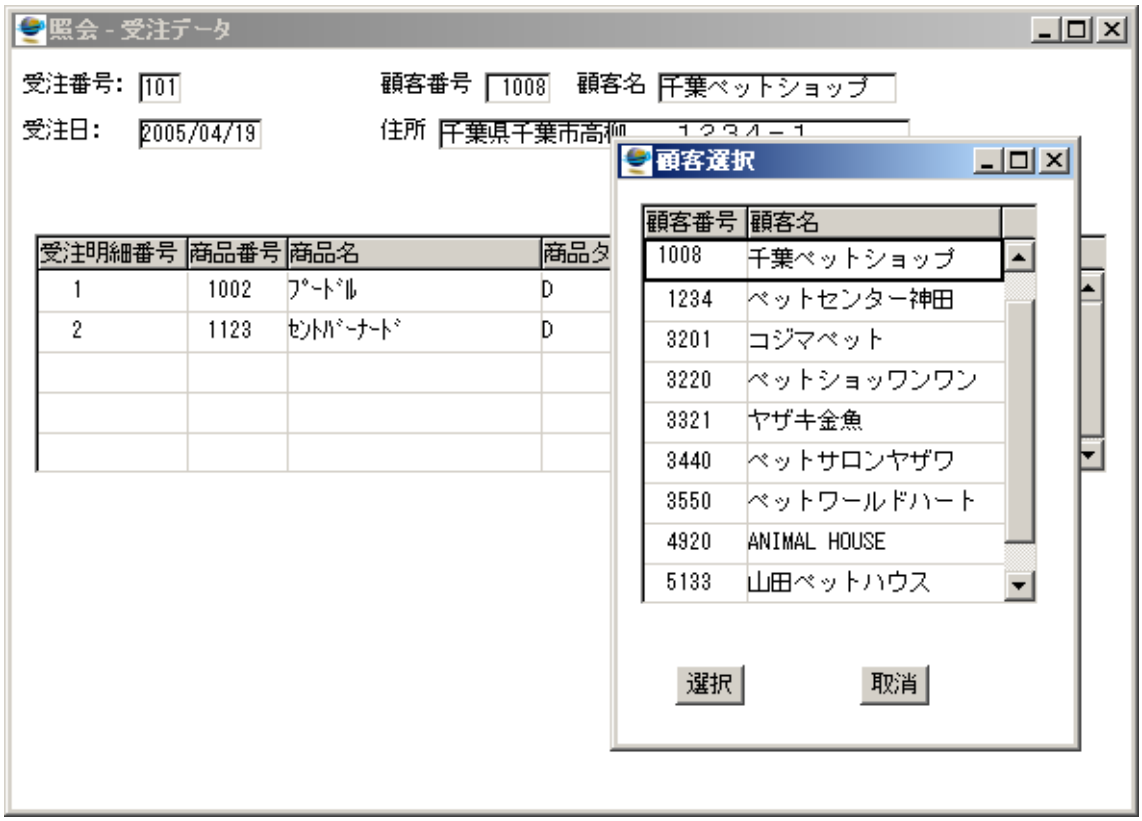


図 11-1 顧客番号の変更

11.1. 注文票の修正

最初に、注文レコードを修正する場合を考えます。注文レコードに対しては、ユーザは次のような修正ができます。

- 受注日の修正
- 顧客番号の修正

その他の項目は、自動的に設定・計算される項目なので、ユーザが手で修正することはできません。
実際の業務アプリケーションでは、このような項目を後から変更するようなことはないかもしれませんが、ここでは Magic の機能を説明するための例として、できることにします。では、修正が起きたときに、どのようなデータ修正の波及があるかを考えてみると、次のようになります。

- 受注日の修正： 今回作成しているアプリケーションに限れば、修正に伴う波及はありません。
- 顧客番号の修正： 顧客番号を変えた場合には、以下の修正が必要です。
  - ✧ 顧客名、住所の表示を更新します。
  - ✧ 顧客マスタの「取引回数」を更新します。
  - ✧ 顧客マスタの「受注累計額」を更新します。

これらの波及に対して、今まで作ってきたロジックでどこまで対応できているかは、次のようになります。

① 顧客名、住所の表示の更新： データリンクが再計算されるので、新たなロジックを作成する必要はありません。
② 顧客マスタの「取引回数」を更新します： これはまだ対応していません。
③ 顧客マスタの「受注累計額」を更新します： 親タスクのレコード後処理で加算モードの項目更新コマンドで差分の更新をおこなっていますが、顧客番号が変わった場合にも、このコマンドが正しく処理します。従って、新たなロジックは必要ありません。

結局、② の取引回数の更新だけが新規に必要になりますが、これも非常に簡単に対応できます。
顧客番号が変わった場合、例えば、1008 千葉ペットショップから 3201 コジマペットに変更になった場合に、それぞれの顧客マスタレコードの取引回数に対して行いたい処理は、

- 変更前の 1008 千葉ペットショップに対しては、－1
- 変更後の 3201 コジマペットに対しては ＋1

です。
これを行うには、やはり加算モードの項目更新コマンドを使いますが、式 としては、定数の 1 とします。そのようにすると、加算モードの項目更新コマンドで、リンクが再計算された場合の規則に則り、次のような計算が行われます。

顧客	変更前の値	変更後の値	差分値
1008	1	リンクからはずれたので、0 とみなされる	－1
3201	リンクされていなかったので、0 とみなされる	1	＋1

表 11-1 取引回数に対する加算モードの項目更新

加算モードの項目更新コマンドは、ここでも便利に利用することができることがわかると思います。
プログラムでは、この項目更新コマンドは親タスクのレコード後処理で行いますが、修正するデータ項目はす

べてレコードメインで予め登録しておかなければならないので、レコードメインで「取引回数」もセレクトコマンドで選択するようにします。

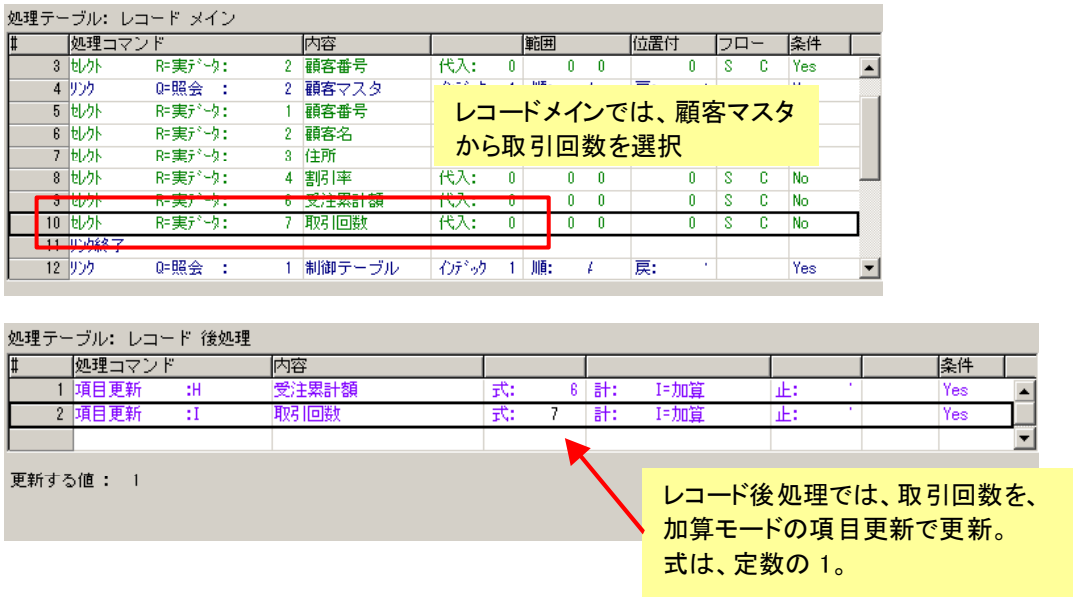


図 11-2 取引回数の更新処理

実際に実行してみましょう。

最初に、顧客マスタのデータを確認しておきます。1008 の千葉ペットショップで受注累計額が 28295 円、取引回数が 1 になっていて、3201 コジマペットではこれがいずれも 0 (空白)になっていることを確認してください。

プログラムリポジトリから、受注入力のプログラムを実行します。注文 101 で、顧客を 1008 から 3201 に変更してみます。顧客番号の変更は、キーボードから番号を直接入力しても構いませんが、ズームすると顧客一覧が出てきますので、そこから選択することもできます。

その後、プログラムを終了し、再度テーブルリポジトリを開き、顧客マスタを APG で開いてください。千葉ペットショップの受注累計額と取引回数がいずれも 0 (空白)になり、かわりに、3201 コジマペットで、受注累計額が 28295 円、取引回数が 1 になっているはずです。

11.2. 新規注文票の作成

今度は、新規に注文票を登録できるようにします。

新規に注文票を作成する場合には、次のことを行わなければなりません。

①. 受注番号を発番する。

②. 受注日をデフォルトで今日に設定する。

以下にどのようにして対応していくかを見ていきます。

11.2.1. 受注番号の発番

受注番号は、制御テーブルの最終受注番号により管理されています。新しく注文レコードを作成する場合には、この値に ＋１して、新注文番号とします。

プログラムでは、次のようにします。

- レコードメインで、制御テーブルをリンクしているところで、カラム３(最終受注番号)も選択するようにします。

図 11-3 レコードメインで最終受注番号を選択

- レコード前処理で、受注番号に、最終受注番号＋１を代入します。ただし、これは登録モードの時だけとします。登録モードの判定は、明細行の場合と同様に、条件式 Stat(0,'C'MODE) で行います。

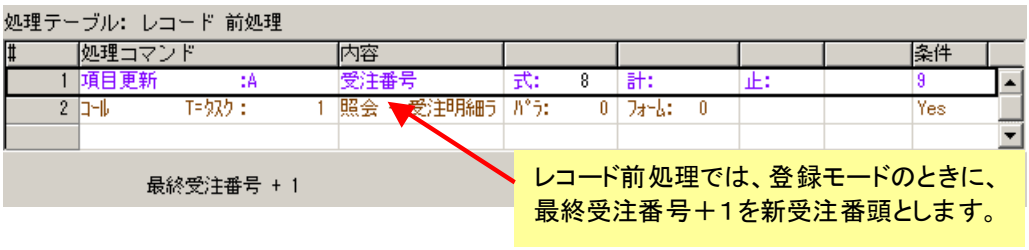


図 11-4 レコード前処理で、新受注番号を設定

- レコード後処理で、最終受注番号を＋１します。これにより、制御テーブルの最終受注番号が更新され、書き込まれます。これも登録モードの場合にだけ行います。

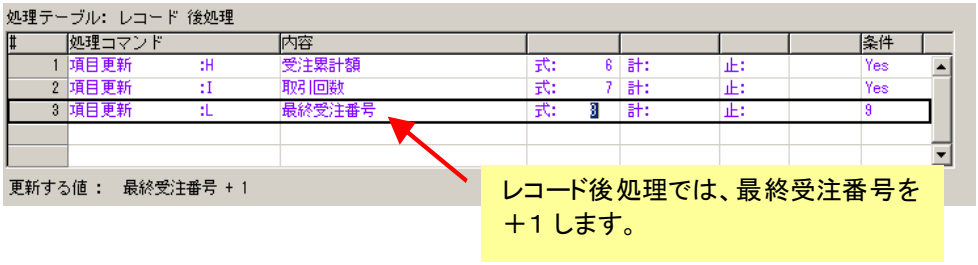


図 11-5 レコード後処理で、最終受注番号を更新

11.2.2. 受注日のデフォルト設定

受注日はデフォルトで今日とします。

登録モード時のデフォルト値の設定は、代入式で行います。受注日をデフォルトで今日とするには、受注日のセレクトコマンドの代入式で、Date() を指定します。この関数は、今日の日付を返すものです。

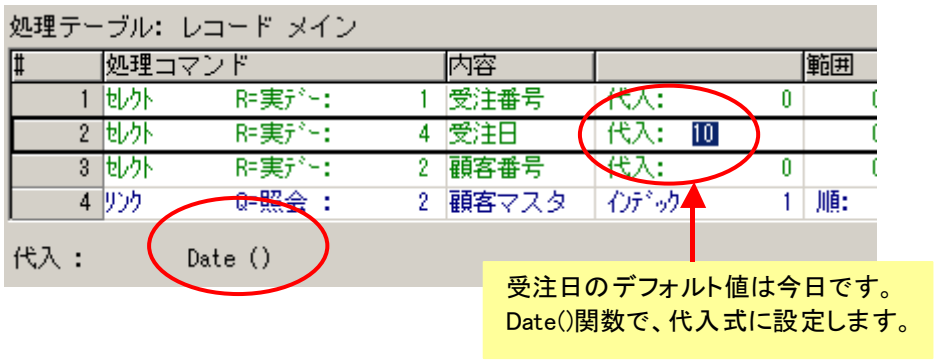


図 11-6 受注日のデフォルトは「今日」にします

11.2.3. 実行してみる

以上で、新規登録への対応は終わりです。再計算、再リンク、マスタへの更新などのロジックは、これまで作ってきたものですべてカバーされています。

実際に実行して確認してみてください。

プログラムを開始し、F4 キーを押せば、登録モードになり、新規受注レコードを入力できるようになります。

確認点としては、以下のようなものがあります。

- 新しい受注番号が、既存のものと重複していないこと。
- 顧客番号を選択した後、顧客名と住所とが正しく表示されていること。
- 明細行を何件か登録して、合計、明細合計額、受注合計額などが正しく計算されていること。
- プログラムを終了し、顧客マスターと商品マスターをそれぞれ APG で開いて、今登録した受注の分だけデータが更新されていること。

11.3. 注文票の削除

受注入力プログラムの最後として、受注票を削除できるようにしてみましょう。

受注データのレコードを削除したい場合、単に「受注データ」テーブルのレコードを削除するだけでは不十分です。なぜなら、受注データの各レコードに対しては、受注明細のレコードが何件か従属しているからで、受注データのレコードを削除したら、それに従属する明細レコードも同時に削除しなければなりません。

これに対応するには、いくつかの方法が考えられますが、今作ったプログラムを大きく変更せずに、一番簡単に対応するには、「明細行が 0 件になるまで、受注レコードの削除を許可しない」というやりかたがあります。つまり、ある受注票を削除しようとしたら、ユーザはまず、明細行を 1 件づつ削除し、全件削除してから、受注レコードを削除する、という手順をすることになります。

明細行の削除については、すでに対応しているので、新たに付け加える必要があるロジックは、「明細行が 0 件の場合にのみ、親タスクでレコード削除を許可する」という条件付けです。ここでは簡単のため、明細行が何件あるかをチェックする代わりに、「受注合計額が 0 円かどうか」で条件付けすることにします。

削除の許可の条件は、タスク制御ダイアログの「削除」パラメータに、式で指定します。

1. 受注入力プログラムを開きます。
2. メニュー 「タスク環境(K)」から「タスク制御(C)」を選びます。(あるいは、Ctrl + C キーを押します)。
3. 「削除」パラメータからズームします。式テーブルが開きます。
4. 式テーブルの最後尾に新規行を作成し、Q=0 (受注合計額 = 0) と設定します。
5. OK ボタンを押すと、式番号が「削除」パラメータ欄に設定されます。
6. タスクを閉じます。

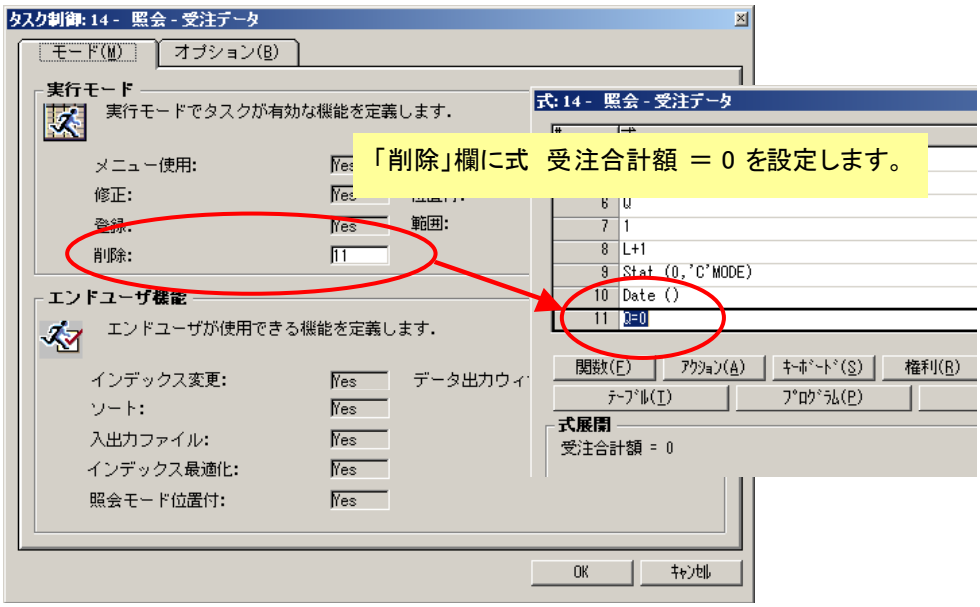


図 11-7 削除許可条件を設定

プログラムの修正は以上です。

11.3.1. 実行してみる

早速実行して、動作を確認してみましょう。

1. 受注入力プログラムを起動します。
2. カーソルが親タスクにある状態で、 F3 キーを押します。何も反応せず、レコードは削除されません。これは、まだ明細行があり、受注合計額が 0 になっていないので、削除許可の条件が成立していないからです。
3. 明細行に入り、 F3 キーによりすべての明細行を削除します。
4. ESC キーで親タスクに戻ります。(マウスで親タスクの項目をクリックしても戻ります)。
5. F3 キーを押します。今度は「削除しますか？」の確認ダイアログが出ます。
6. はい(Y) を押すと、受注票 101 が削除されます。

図 11-8 削除確認のダイアログ

11.4. 次の課題

以上、これで受注入力プログラムはひととおり完成です。
受注入力というのは、一見簡単なようですが、追加・修正・削除のいろいろなパターンと、データの修正による波及の対応とを考えると、意外と複雑なプログラムです。
本章では、Magic で実際にひととおりの動作をするプログラムを作ってみることで、Magic の実行エンジンが提供するさまざまな機能を学んできました。
作りながら、Magic のプログラムの定義というのは、一般のプログラミング言語での手続き志向のものとは異なり、データ間の関係やロジックを定義する、高度な記述性を持ったものであることがおわかりになったと思います。これが、Magic の高い生産性と保守性の秘密のひとつです。

本書では扱いませんでしたが、実際の業務アプリケーションは、複数のユーザが同時にデータベースをアクセスして更新・追加・削除を行うものです。そのため、本格的なアプリケーションでは、マルチユーザ環境でも正しく動作するための考慮が必要になります。
Magic では、マルチユーザ環境でも正しく動作するためのメカニズムも各種備えていますが、今回作成したプログラムでは、例として簡単化するためにマルチユーザ環境での考慮を行っていないため、マルチユーザ環境でスムーズの動作させるためには、もうすこし修正が必要です。
マルチユーザ環境への対応については、弊社の自習書やセミナーなどで扱っているので、ぜひご利用ください。

12. ユーザイベント

以前のバージョンの Magic の実行エンジンはフロー中心の実行モデルにより実行が進められていましたが、V9 になってイベントドリブンの機能が強化されました。これにより、Windows の GUI インターフェースでのプログラム作成が容易になると同時に、強力なイベントハンドリングのメカニズムにより、プログラムの再利用性（従って生産性と保守性)が向上しています。
「ボタンとイベント」の章でもイベントについてふれましたが、そこでは「クローズ(C)」や「選択」など、Magic 実行エンジンで予め動作が定義されている内部イベントだけを利用してきました。
本章では、開発者が自由に定義・利用することのできる「ユーザイベント」について、ユーザイベントの定義方法、プッシュボタンへの割り当て、プログラム中での利用方法、イベントをキャッチして処理を行うイベントハンドラの定義の方法などについて説明します。

本章の例としては、商品の価格変更プログラムを使います。このプログラムは、最終的には次のような機能を持たせます。

- 価格の変更方法として、手動（ひとつづつ個別に指定する)と、自動（増減%を与えて、全商品について一律に変更する)との二つの方法を提供する。
- 最初の画面で、「自動」「手動」二つのボタンを表示する。
- 「手動」を押すと、「商品マスタ更新」プログラムが開くので、ユーザが個別にデータの更新を行うことができる。
- 「自動」を押すと「変更%入力」画面が出るので、%を指定して「実行」ボタンを押すと、全商品についての価格変更を実行する。

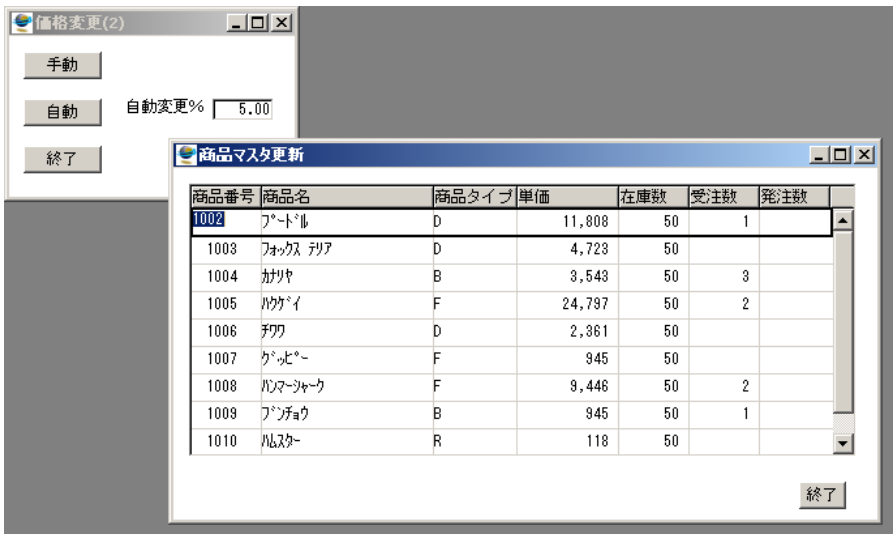


図 12-1 ユーザイベントを用いた価格変更プログラム(最終形)

参考情報： 本章の内容については、第 7 章 「ボタンとイベント」の参考情報の欄も参照してください。

12.1. タスクイベントとグローバルイベント

ユーザイベントには、タスクイベントとグローバルイベントという 2 種類のものがあります。
タスクイベントというのは、特定のタスクのイベントテーブルで定義され、そのタスクおよびサブタスクでだけ有効です。あるタスクに固有なイベントは、タスクイベントとして定義します。
グローバルイベントというのは、「メインプログラム」 (プログラムリポジトリで第 1 番目に定義されているプログラムで、各アプリケーションにはかならず一つ定義されています)のイベントテーブルで定義します。ここで定義されたユーザイベントは、アプリケーション全体で有効であり、モデルリポジトリで定義されるプッシュボタンなどで割り当てることもできます。アプリケーション全体で共通に使われるようなイベントや、モデルに定義して共有化を図りたい場合などには、グローバルイベントとして定義します。
本章では、まず基本としてタスクイベントを 12.3 節 タスクイベントで説明し、グローバルイベントについては、12.4 節 グローバルイベントで説明します。

12.2. ユーザイベント利用の基本形

「クローズ(C)」などの内部イベントは、Magic 実行エンジンで予め動作が決められています。例えば、「クローズ(C)」イベントが発生すると、現在開いているウィンドウが閉じる(タスクが終了する)という動作が実行されます。

それに対しユーザイベントは、Magic 実行エンジンで予め動作が定義されていません。そのためユーザイベントは、それをキャッチして処理を行う**イベントハンドラ**が定義されている必要があります。イベントハンドラが定義されていない場合には、ユーザイベントが発生しても、Magic 実行エンジンは何もせず、単にイベントは捨てられます。

従って、ユーザイベントを定義利用して GUI 画面の操作をコントロールするプログラムは、次のような一般形を持っているのが普通です（図 12-2)。

- ユーザイベントを必要なだけ、イベントテーブルで定義する。
- プッシュボタンのコントロール特性で、ユーザイベントを割り当てる。このようなプッシュボタンが実行時にユーザによりクリックされたときには、割り当てられたユーザイベントが発生します。
- イベントをキャッチするためのイベントハンドラを定義する。このイベントハンドラの中で、必要な処理(例えば、商品マスタ更新プログラムをコールコマンドで呼び出すなど)を行います。

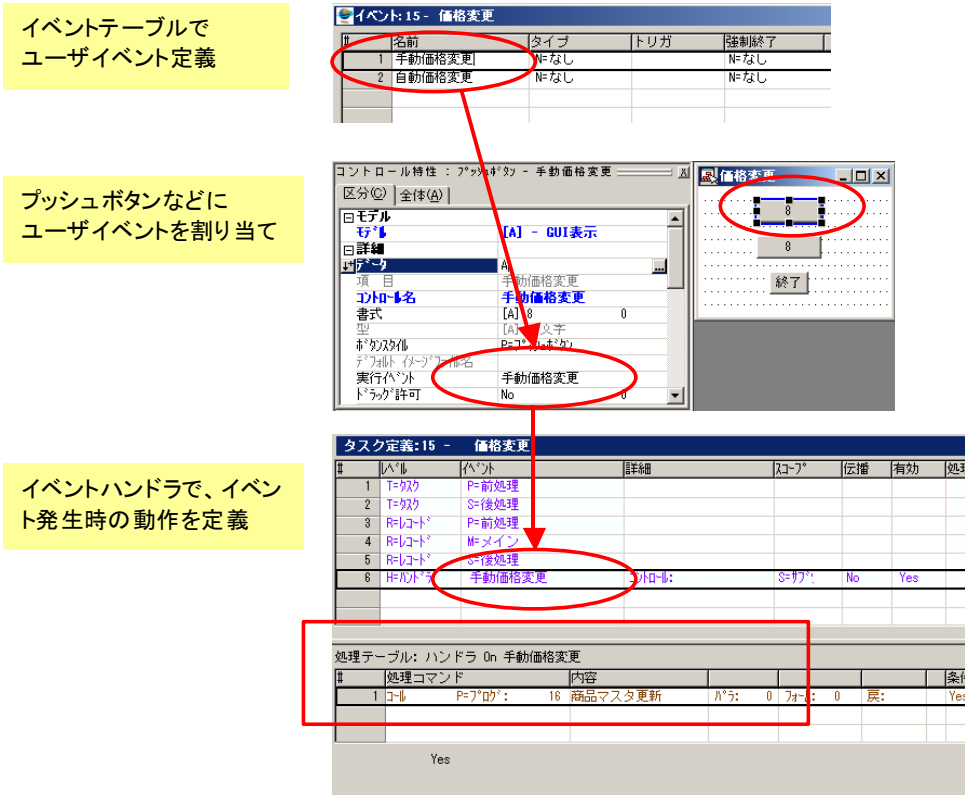


図 12-2 ユーザイベント利用法の基本形

12.3. タスクイベント

ユーザイベントの基礎として、タスクレベルで定義する**タスクイベント**をまず説明します。タスクイベントというのは、ユーザが定義するユーザイベントの一種ですが、定義したタスクおよびそのサブタスクでだけ有効なものです。

12.3.1. ユーザイベントとプッシュボタンの作成

まず最初に、新規タスクを作成し、タスク特性を設定します。

1. プログラムリポジトリで、最下行で F4 キーを押し、新規プログラムを作成します。
2. F5 キーを押すと、タスク特性が開きます。
3. 名前は「価格変更」とし、その他はそのままにします。

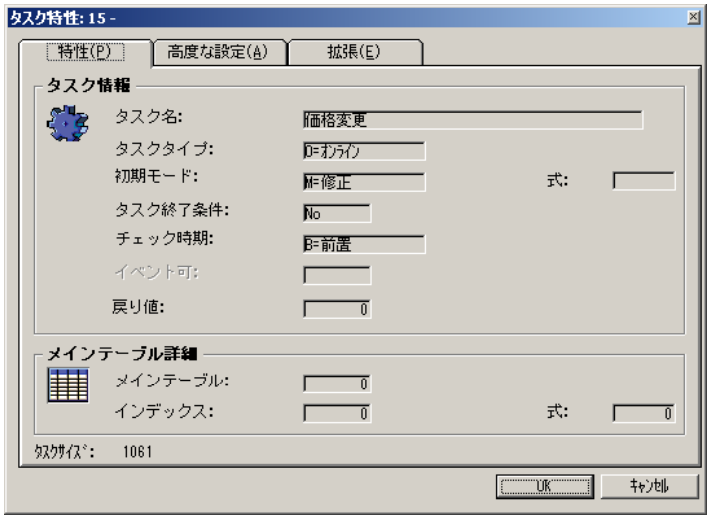


図 12-3 タスク特性

参考： 今まで出てきたタスクはすべてメインテーブルを指定していましたが、今回作るタスクでは、テーブルを使わないでただメニューボタンを表示するだけなので、メインテーブルは 0 のままとします。

次に、ユーザイベント「手動価格変更」と「自動価格変更」を定義します。

1. メニュー 「タスク環境(K)」から「ユーザイベント(U)」を選び、イベントテーブルを開きます。Ctrl ＋ K でも同じです。
2. F4 で新しい行を作成します。
3. 名前は「手動価格変更」、「タイプ」は「N=なし」、「強制終了」は「N=なし」とします。
4. 同様に、「自動価格変更」を定義します。
5. Enter で、イベントテーブルを閉じます。

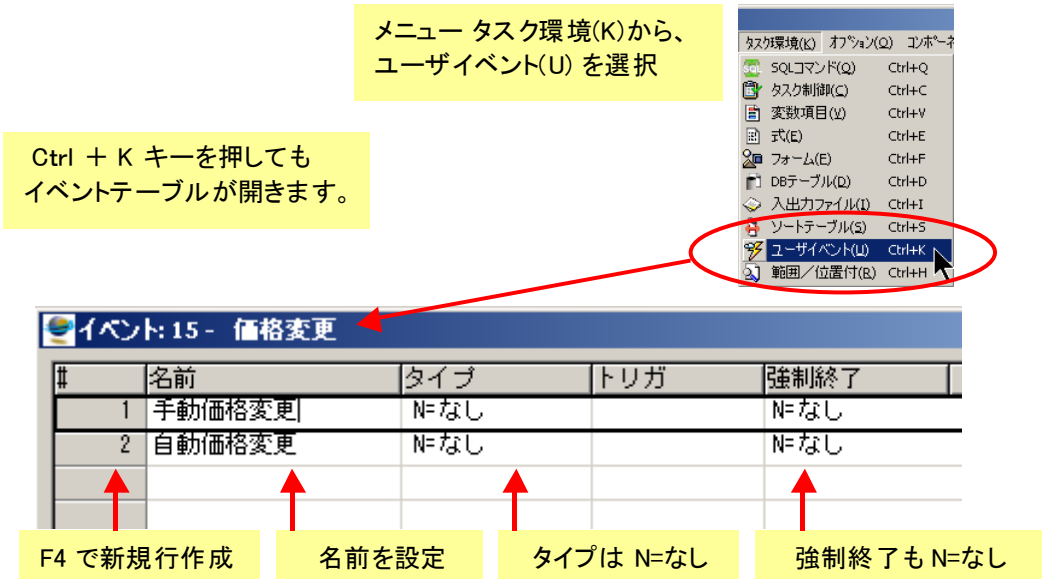


図 12-4 ユーザイベントの定義

レコードメインで、変数「手動価格変更」を定義し、「手動価格変更」イベントを発行するプッシュボタンを関連付けます。

1. レコードメインを開きます。
2. F4 で新規行を作成し、セレクトコマンドを選びます。
3. タイプは「V=変数」(デフォルト)のままとします。
4. F5 でズームして、変数テーブルを開きます。
5. 名前は、「手動価格変更」、型は「A=文字型」とします。
6. Ctrl+P を押します。「ローカル変数特性」のプロパティシートが開きます。
7. 書式は「8」、デフォルト値は「手動」とします。
8. 「GUI 表示」を「プッシュボタン」とします。
9. さらに「GUI 表示」からズームし、「コントロール特性： プッシュボタン」を開きます。
10. 「実行イベント」欄から F5 でズームします。
11. 「イベント」ダイアログで、イベントタイプは「U=ユーザ」とします。
12. イベント欄からズームすると、「イベント一覧」が表示されます。
13. 「手動価格変更」イベントを選択します。
14. OK を押して、「イベント」ダイアログを閉じます。
15. X を押して、「コントロール特性： プッシュボタン」を閉じます。
16. さらに Enter を押して、変数テーブルに戻ります。

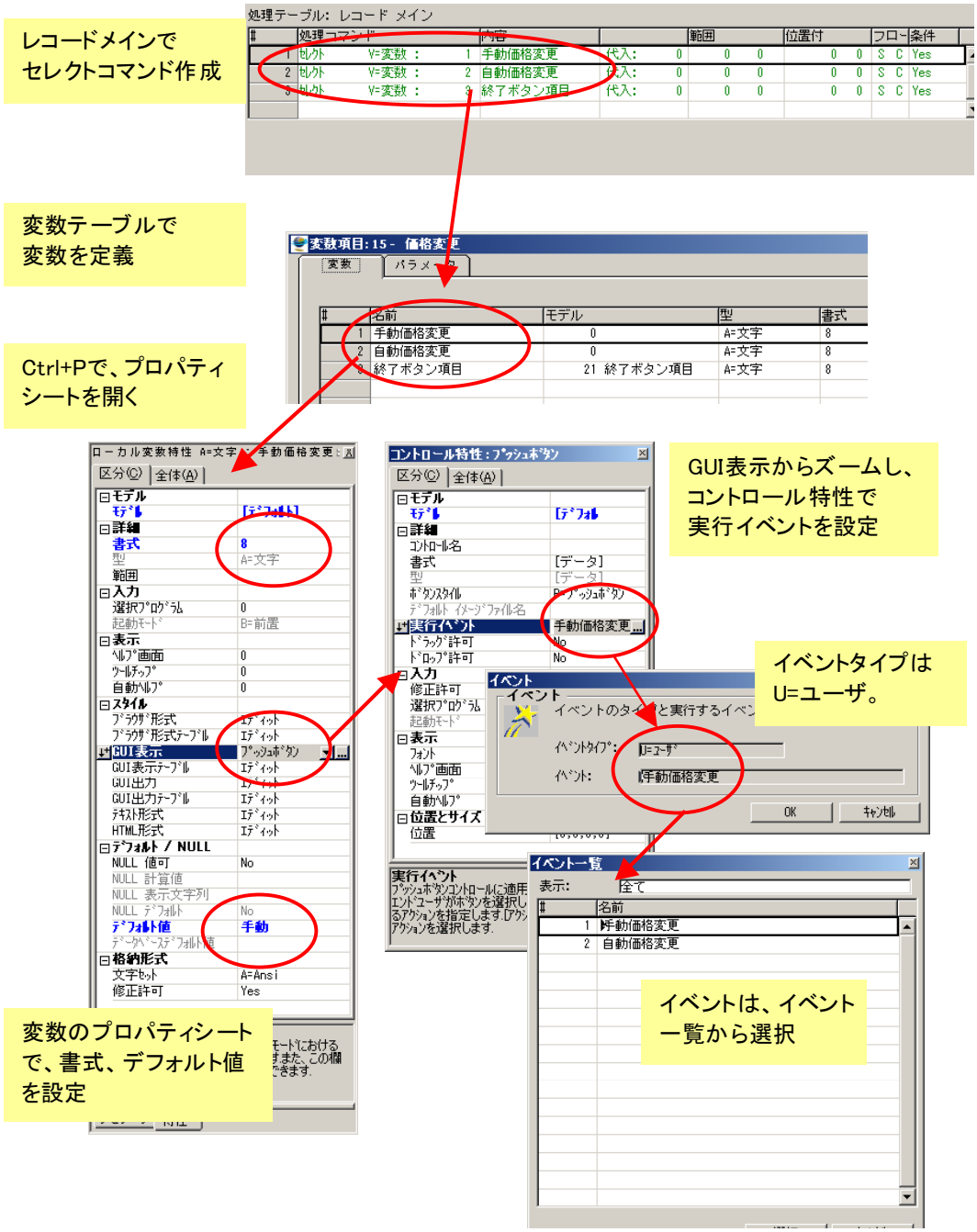


図 12-5 変数にプッシュボタンとユーザイベントを定義

「自動」プッシュボタンのための変数を定義

同様にして、「自動」プッシュボタンのための変数も定義します。

この変数では、名前は「自動価格変更」、デフォルト値は「自動」、プッシュボタンに関連づけるイベントは「自動価格変更」とします。

「終了」ボタン用の変数の定義

ついでに、「終了」用の変数も定義しておきましょう。これは、イベントの章で作成した、「終了ボタン項目」モデルを参照して、簡単に定義できます。

フォームを作成

フォームを作成します。このタスクでは、「手動」「自動」「終了」のボタンを表示するだけです。

1. フォームテーブルを開きます。(メニュー「タスク環境(K)」から「フォーム(F)」、あるいは、 Ctrl +F キーで開きます)。
2. フォームテーブル上で、 Ctrl + G キーを押して、フォームの APG を行います。
3. 表示モードは「S=スクリーンモード」、フォームサイズは「C=表示内容に依存」とします。
4. ズームして、フォームエディッタを開きます。
5. ラベルが余計なので、削除してください。またボタンの配置を移動して、真ん中に表示されるようにしてください。

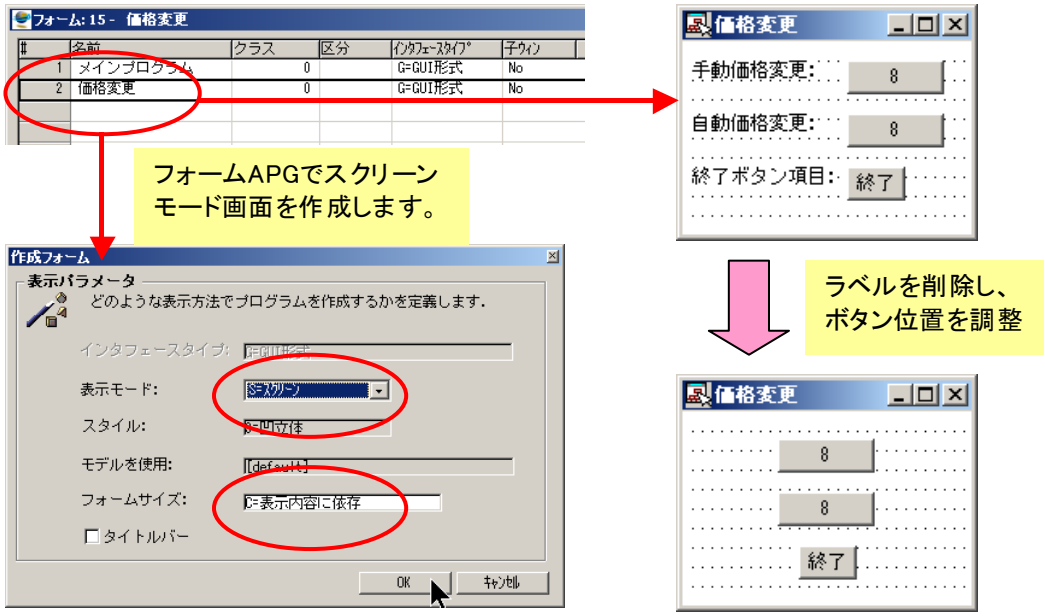


図 12-6 フォームの作成

ここで、配置されているプッシュボタンに、イベントが正しく割り当てられているかを確認しましょう。フォームエディッタを開き、ボタンを選んでください。「コントロール特性」で、表 12-1 に示すようなコントロール特性値が設定されているはずです。

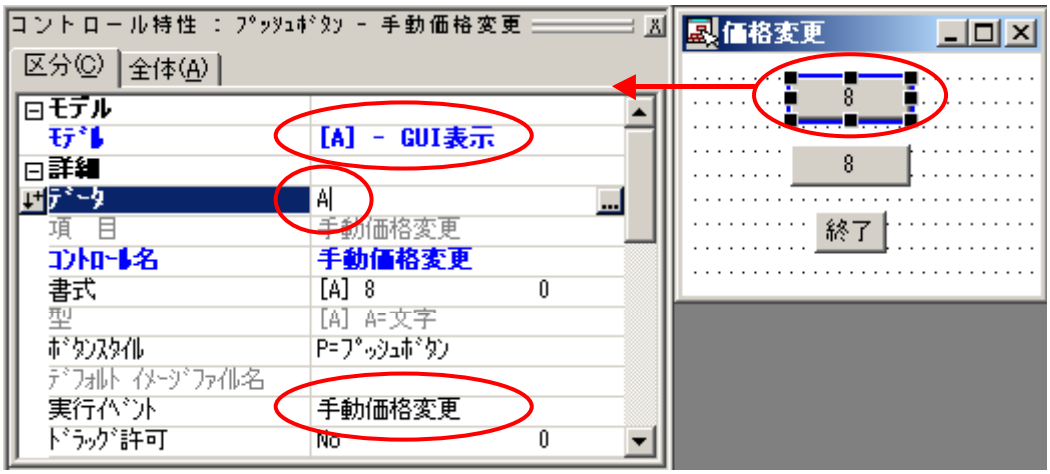


図 12-7 ボタンのコントロール特性

ボタン(上から)	モデル	データ	実行イベント
1	[A] – GUI 表示	A	手動価格変更
2	[B] – GUI 表示	B	自動価格変更
3	[C] – GUI 表示	C	クローズ(C)

表 12-1 ボタンのコントロール特性値

参考： 「モデル」が「[A] – GUI 表示」となっているのは、項目 A (変数テーブルで定義した、「手動価格変更」変数) からプロパティを継承していることを意味します。実行イベントがデフォルトで「手動価格変更」になっているのも、この変数でそのように定義してあるのが継承されているためです。

12.3.2. ユーザイベントのハンドラを作成

ユーザイベントを定義しプッシュボタンを配置しただけでは、ユーザがプッシュボタンを押しても、プログラムは何もしません。プログラムに何かをさせるには、イベントに対応した**イベントハンドラ**を定義する必要があります。

まず最初に、「手動」ボタンを押したときに、「商品マスタ更新」プログラムを呼び出すイベントハンドラを作成してみます。

(1) 商品更新プログラムの作成

価格変更プログラムでイベントハンドラを作成する前に、商品マスタ更新プログラムをまず作っておかなければなりませんので、ここで APG で簡単に作ってやります。

1. プログラムリポジトリの最下行で F4 で新規プログラムを作成します。
2. Ctrl + G キーで、APG を起動します。
3. メインテーブルは、3 (商品マスタ) とします。
4. OK ボタンを押すと、プログラムが作成されます。
5. ズームして開きます。
6. タスク特性を開きます。(メニュー 「編集(E)」から「特性(P)」を選ぶか、あるいは、Ctrl + P キーで開きます)。
7. タスク名は「商品マスタ更新」、初期モードは 「M=修正」にします。

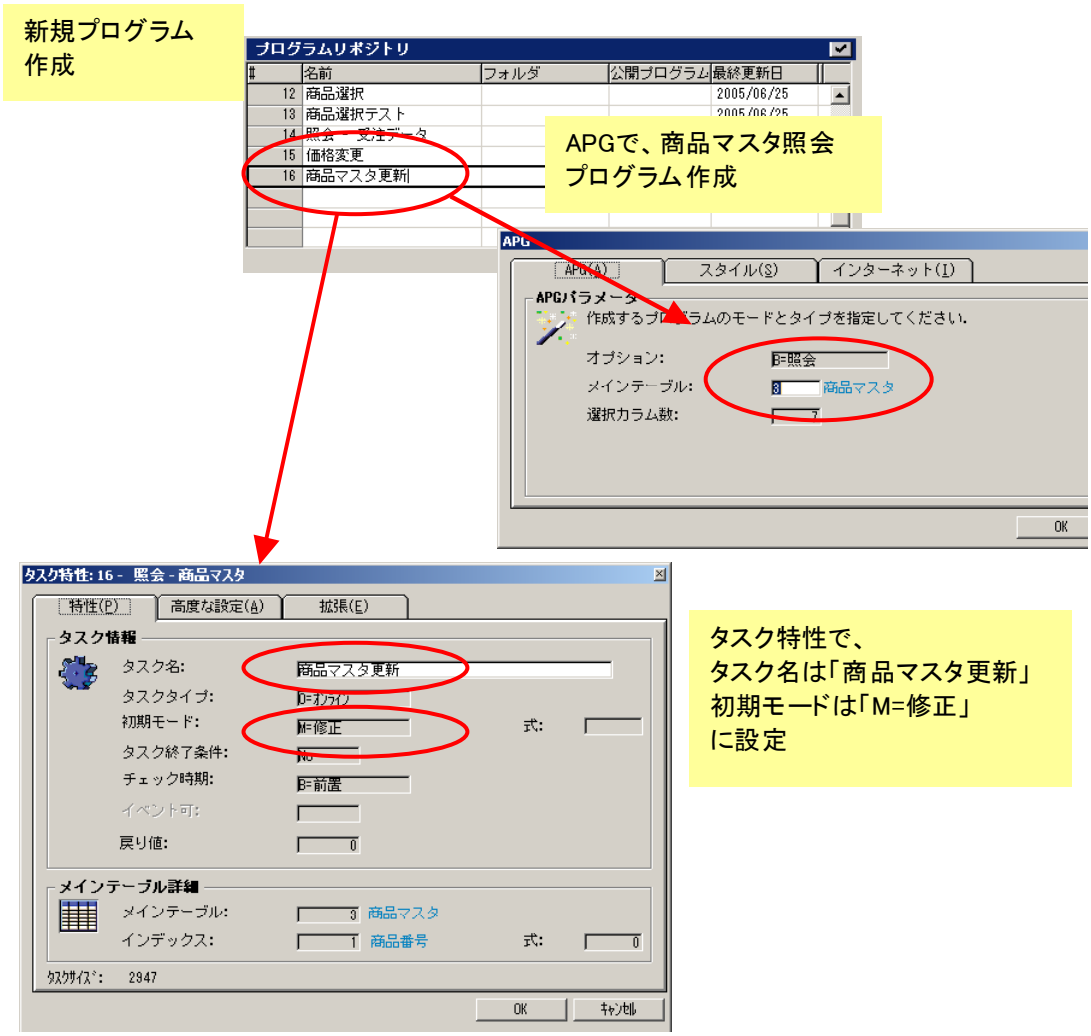


図 12-8 商品マスタ更新プログラムの作成

参考： 便利のため、「終了」ボタンも作っておくと良いでしょう。

イベントハンドラの定義

続いて、価格変更タスクでイベントハンドラを定義し、今作った商品マスタ更新プログラムを呼び出すようにします。

1. 「価格変更」タスクを開く。
2. 「手動価格更新」のイベントハンドラを作成：
   ① 「タスク定義」のレベルテーブルで、マウスで最終行（「R=レコード」「S=後処理」）をクリックする。
   ② F4 で新しい行（ハンドラ）を作成。
   ③ 「レベル」＝「H=ハンドラ」とする。
   ④ 「イベント」欄で、F5 でズームする。→ 「イベント」ダイアログが開く。
   ⑤ 「イベントタイプ」を「U=ユーザ」とする。
   ⑥ 「イベント」欄で F5 でズームする。→ 「イベント一覧」が開く。
   ⑦ 「手動価格更新」にカーソルを合わせ、「選択」をクリック。→ 「イベント」ダイアログに戻り、「イベント」が「手動価格設定」に設定されている。
   ⑧ OK を押して、タスク定義画面に戻る。

3. マウスで「処理テーブル： ハンドラ On 自動価格変更」をクリックする。→ カーソルが「処理テーブル」のタイトル行に移る。
4. ハンドルの中では、コールコマンドで、「商品テーブル更新」プログラムを呼び出すようにする。
   ① F4 で新しい行を作成する。
   ② 「コール」「P=プログラム」
   ③ でズームすると、プログラム一覧が表示されるので、「商品テーブル更新」を選択する。→ プログラム番号が設定される。
5. Enter でタスクを閉じます。

このようにイベントハンドラを定義しておいてやれば、「イベント」欄に定義したイベント(この場合、「手動価格更新」ユーザイベント)が発生した場合、このイベントハンドラがキャッチして、処理テーブルで定義されているコマンドを実行します。ここでは「商品マスタ更新」プログラムを呼び出すコールコマンドが定義されているので、商品マスタ更新プログラムが呼び出されます。

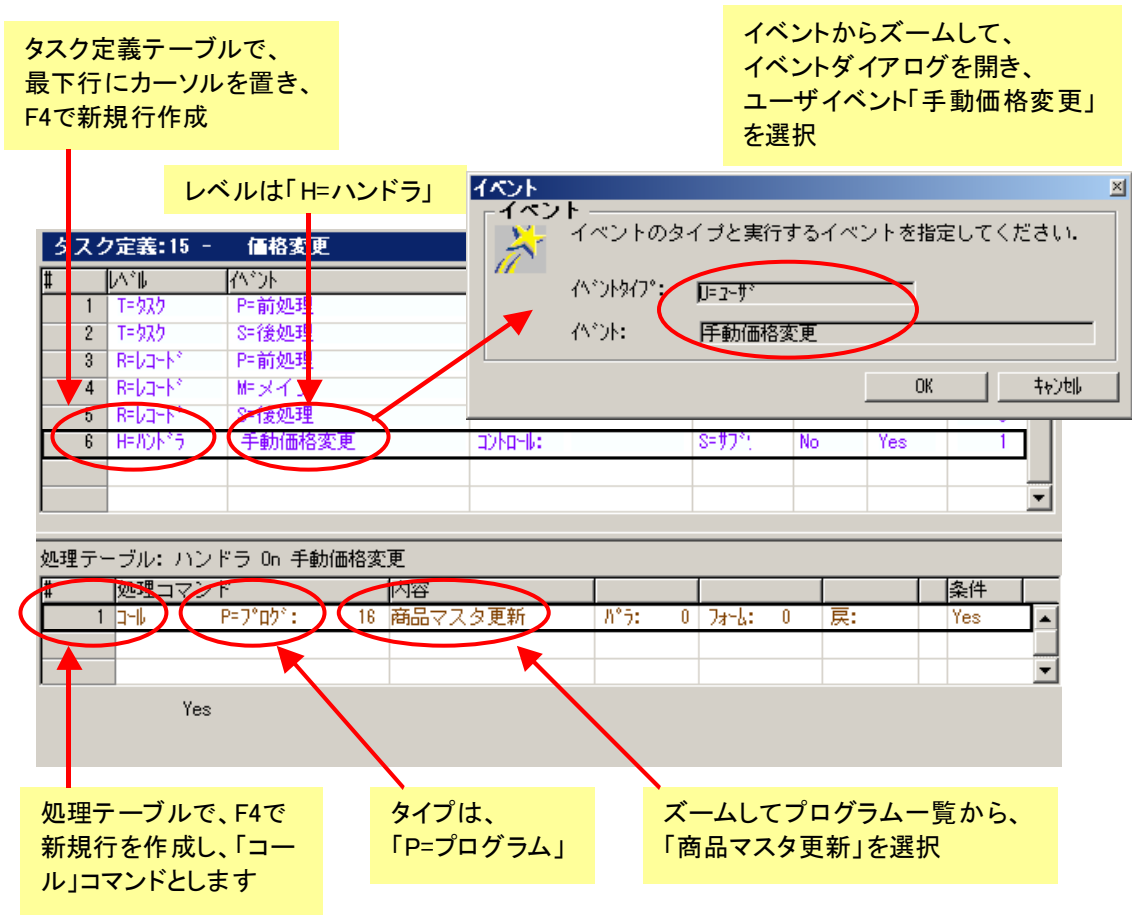


図 12-9 イベントハンドラの定義

12.3.3. 実行してみる

プログラムリポジトリから、「価格変更」プログラムを F7 キーで実行します。 手動 ボタンを押したら、商品マスタ更新プログラムが開くことを確認してください。

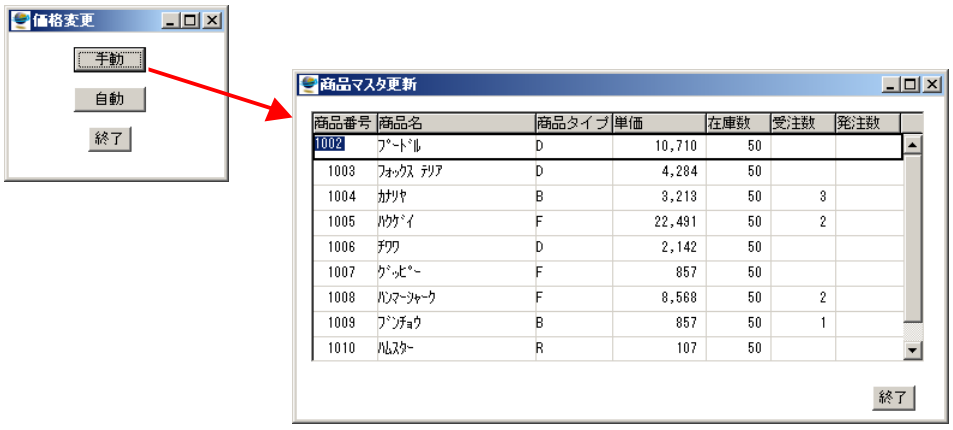


図 12-10 商品マスタ更新プログラムの呼び出し

12.4. グローバルイベント

グローバルイベントとは、ユーザイベントの一種ですが、タスクイベントが定義されているタスクとそのサブタスクでだけ有効なのに対し、グローバルイベントはアプリケーション全体で有効であるところが異なります。グローバルイベントは、モデルリポジトリでプッシュボタンなどのモデルと組み合わせて使うと、最もその効果を発揮できます。

前節では、フォーム上のボタンを定義するのに、変数を定義し、各変数にプッシュボタンを割り当て、そのプッシュボタンにユーザイベントを割り当てた上で、始めてプログラム中で利用できるようになりました。しかし、ボタンごとにこのような定義をしているのは煩雑です。

そこで、例えば「実行」という名前のグローバルイベントを定義し、それを発生させるプッシュボタンと、そのプッシュボタンを割り当てた項目モデルをモデルリポジトリで定義します。

これを使えば、プログラムで変数を定義する際にこの項目モデルを参照するだけで、「実行」イベントを発行するプッシュボタンを備えた文字型データ項目を簡単に定義することができるようになり、多くのボタンを配置したプログラムであっても、作成の手間がかなり省けます。

12.4.1. グローバルイベントの定義方法

グローバルイベントは、メインプログラムのイベントテーブルで定義します。例として、「実行」という名前のグローバルイベントを定義します。

1. プログラムリポジトリを開きます。
2. 第１番目の「メインプログラム」にカーソルを置いてズームし、メインプログラムを開きます。
3. イベントテーブルを開きます。 （メニュー「タスク環境(K)」から「ユーザイベント(U)」選択、あるいは Ctrl + K キーで開きます。）
4. 名前を「実行」、タイプ＝「N=なし」、強制終了＝「N=なし」とします。
5. タスクを閉じます。

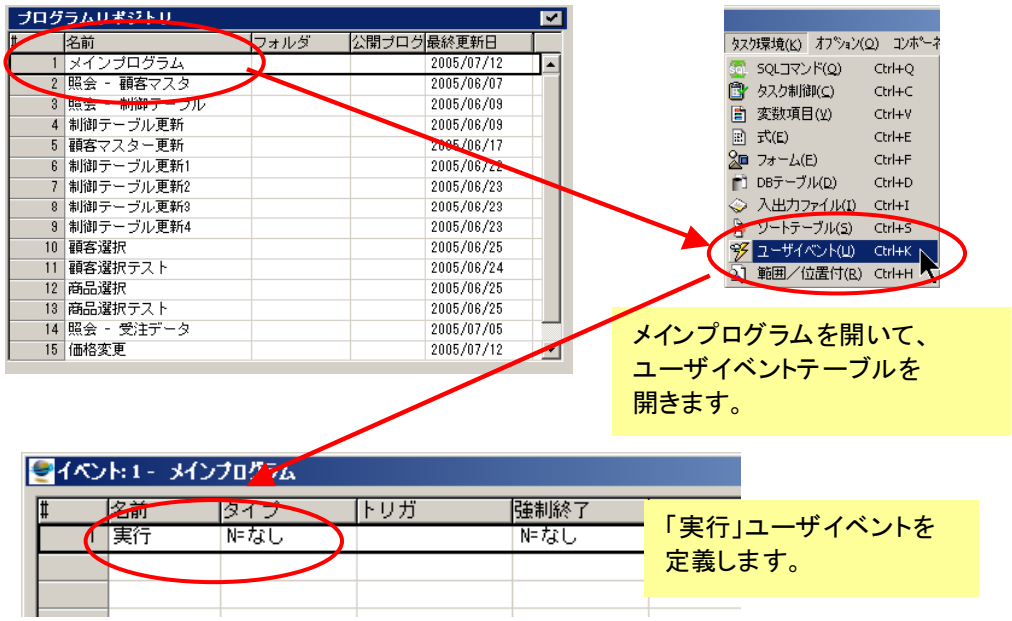


図 12-11 グローバルイベントの定義

12.4.2. プッシュボタンモデルの定義

この「実行」イベントを発行するプッシュボタンをアプリケーション全体で簡単に利用できるように、モデルとしてモデルリポジトリに登録します。

1. 新規にプッシュボタンのモデルを作成します。
   ① モデルリポジトリを開きます。
   ② 最下行で F4 キーを押します。新しいモデルが作成される。
   ③ 名前は「実行ボタン」、クラスは「D=GUI 表示」、型は「P=プッシュボタン」 とします。
2. クリックされたときに、「実行」イベントを発行するようにします。
   ① プロパティシート上の、「実行イベント」にカーソルを移動します。
   ② ズームすると、「イベント」ダイアログが開きます。
   ③ イベントタイプを「U=ユーザ」 とします。
   ④ イベント欄でズームすると、イベント一覧が開きます。 ここに表示されるのは、グローバルイベントだけです。
   ⑤ 「実行」イベントを選択します。
   ⑥ OK を押して、イベントダイアログを閉じます。
3. Enter を押すと、モデルリポジトリに戻ります。

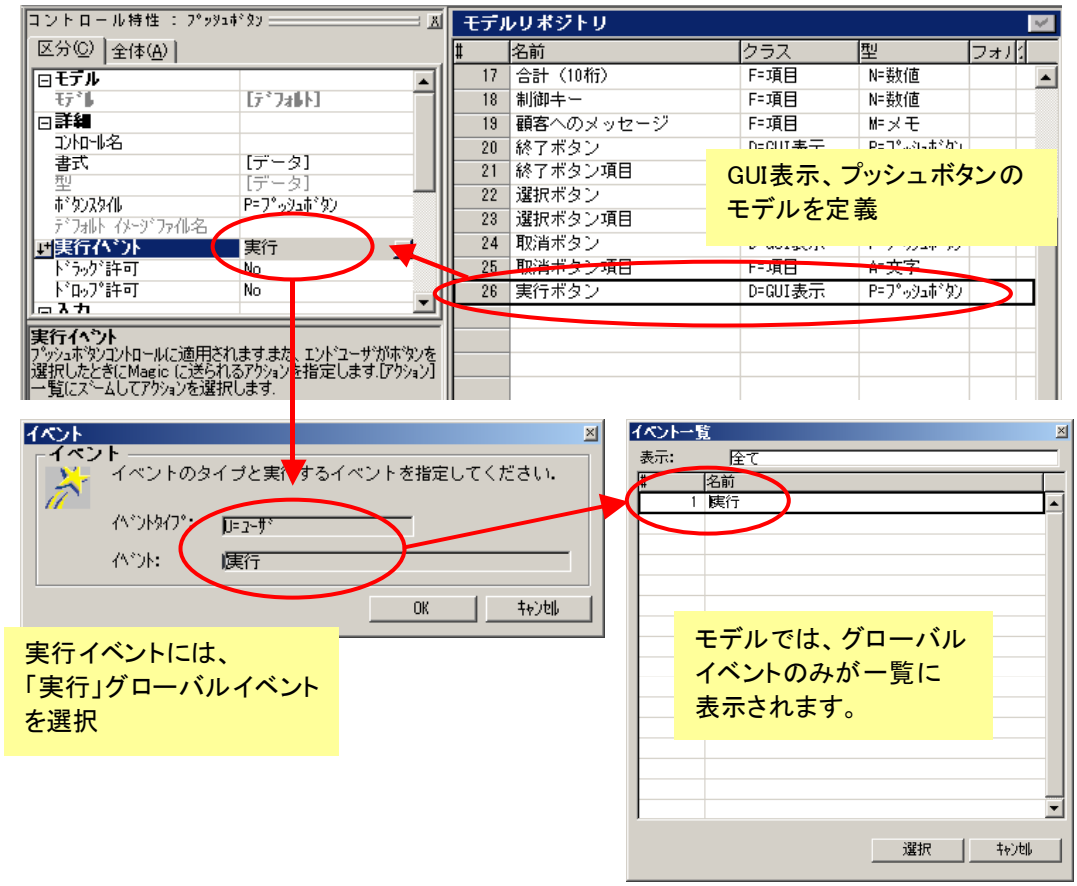


図 12-12 グローバルイベントを参照してプッシュボタンモデルを登録

12.4.3. プッシュボタンに関連づけられた項目モデルを定義

さらに、このプッシュボタンに関連づけられた文字型の項目モデルを定義します。

1. 新規に項目モデルを作成します。
   ① モデルリポジトリを開きます。
   ② 最下行で F4 キーを押し、新しいモデルを作成します。
   ③ 名前は「実行ボタン項目」、クラスは「F=項目」、型は「A=文字」とします。
   ④ プロパティシートで、書式を「8」にします。
2. 「実行ボタン」プッシュボタンモデルを関連づけます。
   ① プロパティシート上の「GUI 表示」に移動します。
   ② ドロップダウンメニューから、「プッシュボタン」を選びます。
   ③ ズームして、コントロール特性を開きます。
   ④ 「モデル」欄でズームすると、 登録されているプッシュボタンのモデルが一覧表示されます。
   ⑤ 「実行ボタン」を選択します。
   ⑥ 「項目特性」のプロパティシートに戻ります。(ウィンドウ右上の X ボタンを押すか、あるいはマウスクリックで戻ります。)
3. モデルリポジトリに戻ります。(Enter 、 ESC 、あるいはマウスクリック)

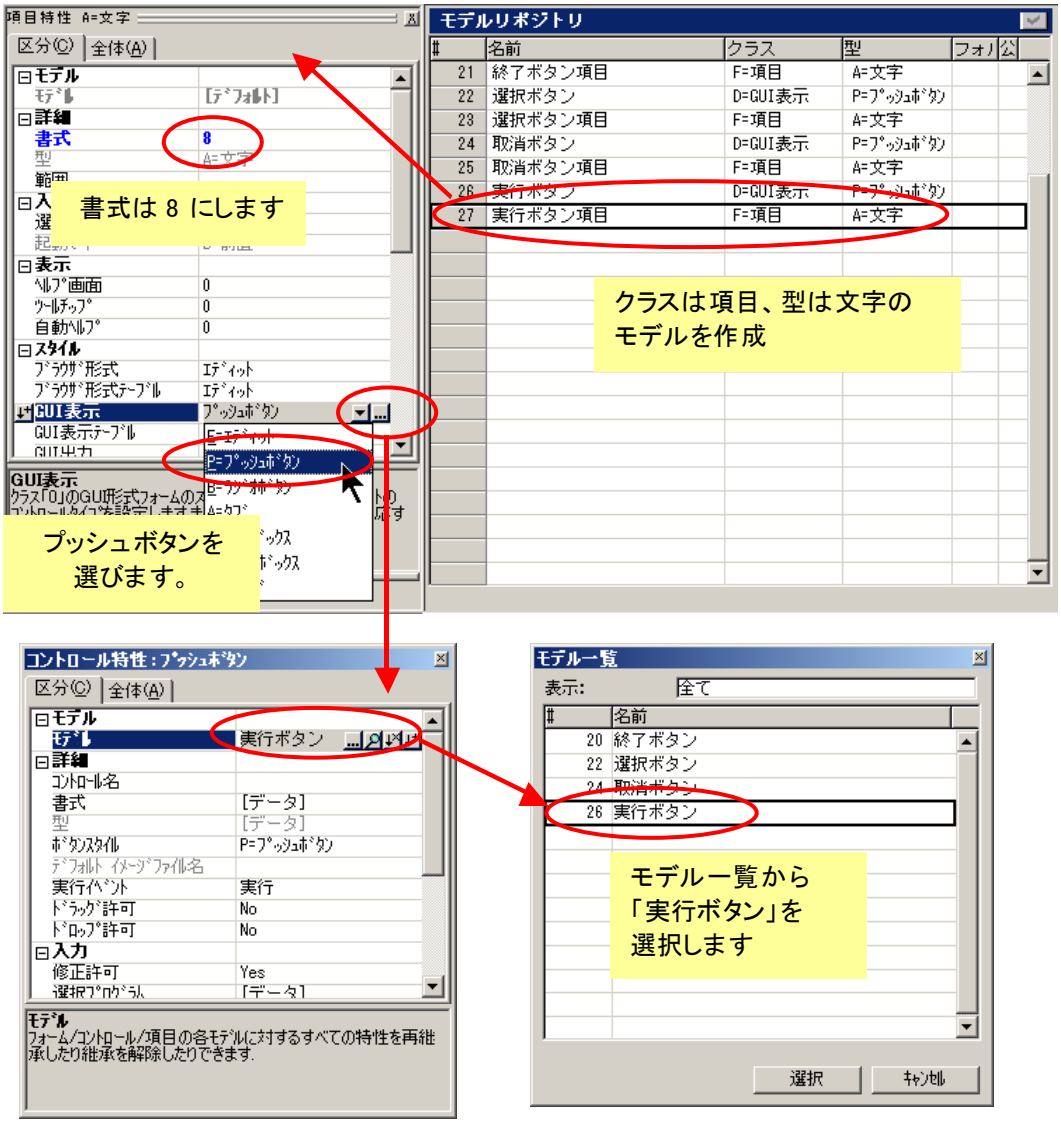


図 12-13 実行ボタンモデルを参照する項目モデルの登録

12.4.4. モデルをプログラムで使う

前節では「価格変更」プログラムを作成するのにタスクイベントを使いましたが、ここでは上で定義したモデルを参照し、グローバルの「実行」イベントを使って最初から再度作成してみます。

1. プログラムリポジトリで新しいプログラムを作成し、開いて、タスク特性で、名前を「価格変更(2)」とします。
2. レコードメインで「手動価格変更」のための変数を定義します。
   ① レコードメインを開き、二つの「セレクト V=変数」コマンドを作成します。
   ② ズームして、変数テーブルを開きます。
   ③ 変数 1、 2 とも、「モデル」欄からズームし、モデル一覧から「実行ボタン項目」を選択します。
   ④ 「名前」欄が「実行ボタン項目」となっているので、それぞれ「手動価格変更」「自動価格変更」とします。
   ⑤ ボタンに表示するデータを特性シートの「デフォルト値」で、それぞれ「手動」「自動」と設定

します。

⑥　レコードメインに戻ります。

3. ついでに、「終了ボタン項目」モデルを参照する「終了」の変数もレコードメインに作成してください。

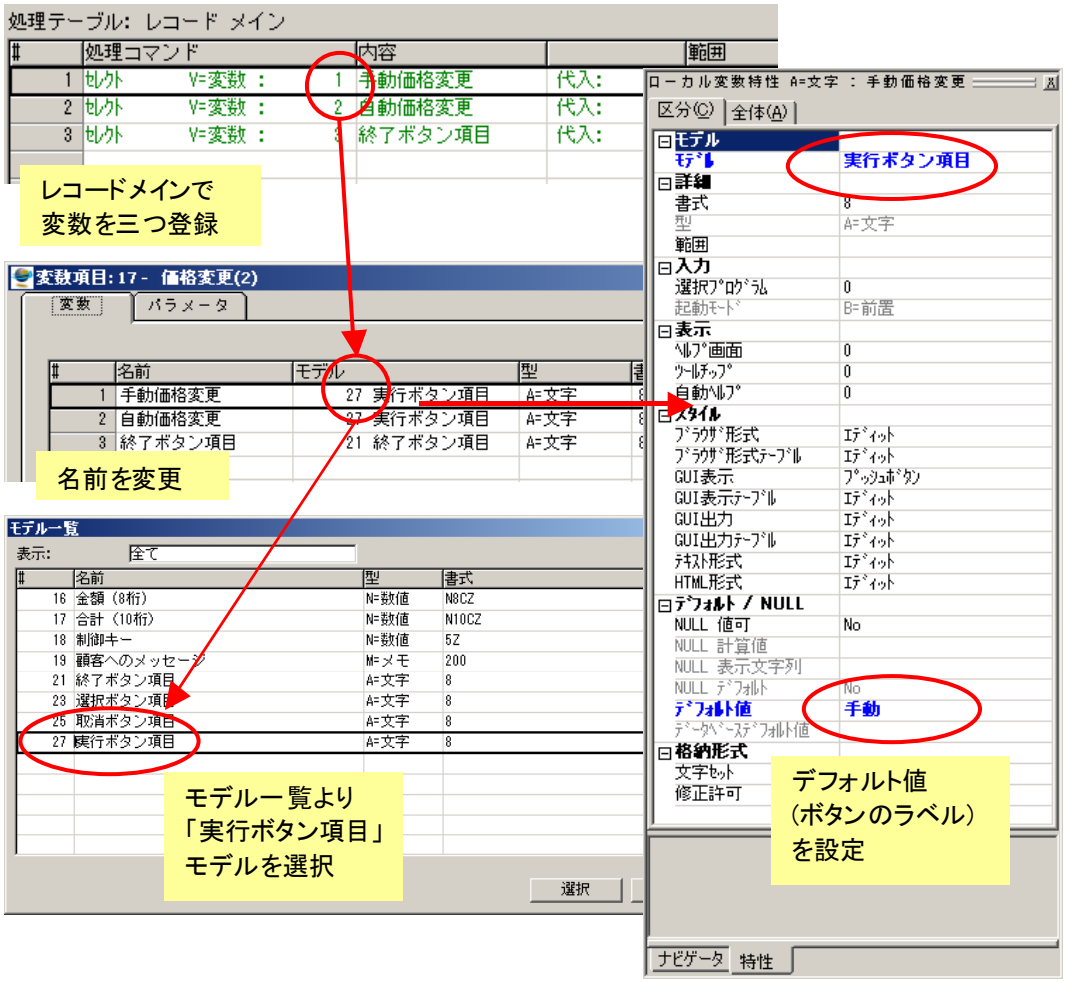


図 12-14 「実行ボタン項目」モデルを参照する変数を定義

4. フォームにボタンを貼り付けます。前節と同様の要領で、図 12-15 のようにフォームを作成してください。

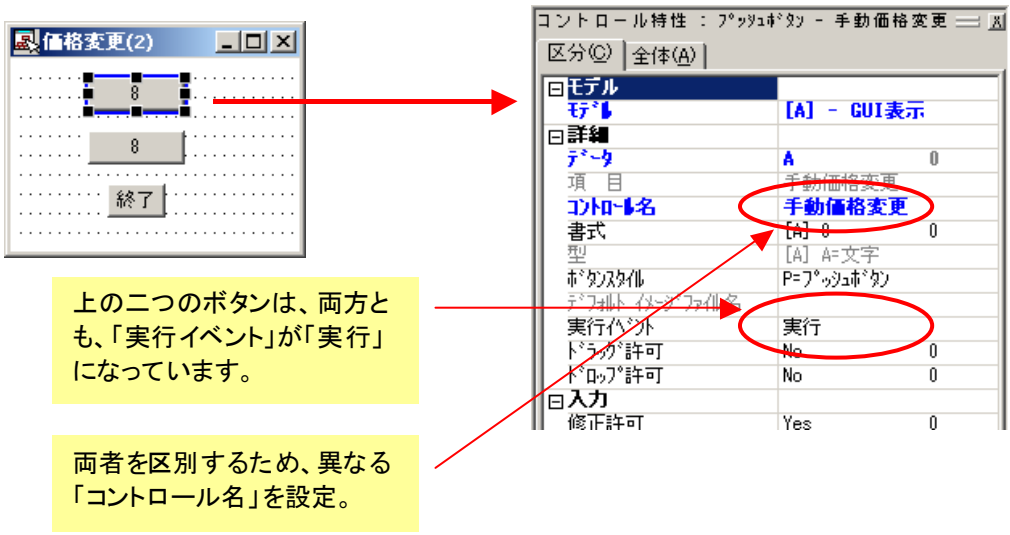


図 12-15 フォームとボタンの特性

ここでフォーム上で、プッシュボタンのコントロール特性で、「実行イベント」を見てみてください。上二つのプッシュボタンの実行イベントは、「実行」というグローバルイベントがモデルから継承されているはずです。

このような設定になっているとき、1 番目のボタンでも 2 番目のボタンでも、クリックすると「実行」イベントが発生します。イベントハンドラでは、この両者を区別できなければなりません。そのために、「コントロール名」を使います。コントロール特性の「コントロール特性」をもう一度見てみてください。コントロール名には、デフォルトで、変数の名前が設定されます。ここでは 1 番目のボタンのコントロール名は「手動価格変更」となっており、2 番目は「自動価格変更」となっています。

では、「手動価格変更」と「自動価格変更」のイベントハンドラを定義しましょう。

1. 「タスク定義」画面に戻り、最下行にハンドラを 1 行追加します。
2. レベルは「H=ハンドラ」、「イベント」は「U=ユーザイベント」で「実行」に設定します。
3. 「詳細」の「コントロール」欄でズームします。
4. コントロール一覧が表示されるので、「手動価格変更」を選択します。
5. 処理テーブルに移り、新規行を作成して、次のようなコールコマンドを登録してください。

   コール　P=プログラム　16　「商品ファイル更新」
6. 同様にして、「自動価格変更」のイベントハンドラも定義します。ただし、こちらの方は、処理テーブルの中身がまだ空のままです。

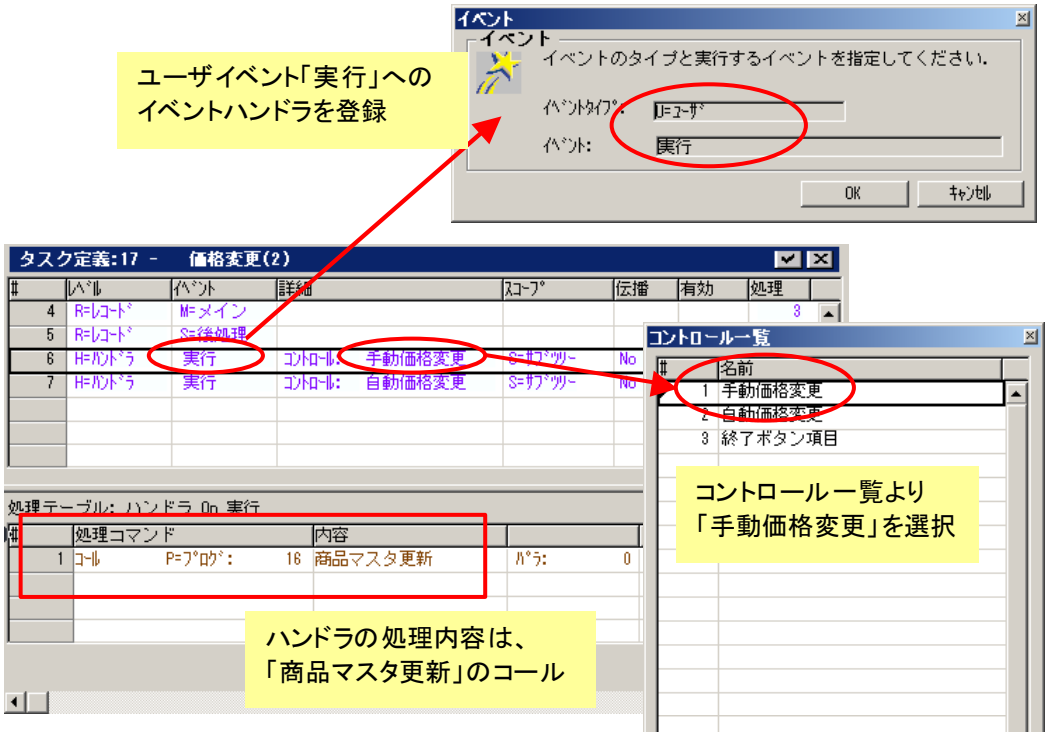


図 12-16 コントロール名を指定して「実行」イベントハンドラを登録

この場合、「実行」イベントハンドラが二つありますが、「コントロール：」でコントロール名を指定しているので、いずれのボタンがクリックされて発生した「実行」イベントなのかを区別できるようになります。

以上で出来上がりです。F8でチェックして、エラーがないか確認してください。

12.4.5. 実行して確認

作成したプログラムの動作を、実際に実行して確認しましょう。

1. 「手動」をクリックしてみてください。「商品ファイル更新」プログラムが開きます。
2. 「自動」をクリックしてみてください。イベントハンドラが空なので、何も起こらないはずです。

12.4.6. 次には・・・

手動価格変更はこれでできましたが、自動価格変更を次章で作るようにしましょう。

13. バッチタスク

今までは作ってきたプログラムはすべて、ユーザにデータ表示をしたり、ユーザの対話的操作(ボタンのクリック、キー入力など)のあるオンラインタスクばかりでした。
Magic ではこのような形式のほかに、**バッチタスク**という形式のタスクがあります。バッチタスクというのは、ユーザの入力待ちがなく、対象となるレコードすべてについて、決められたステップを実行するタスクです。テキスト形式のデータファイルの入出力、印刷、集計、コピーや削除などのデータ保守を行うタスクなどでバッチタスクがよく使われます。
ここでは、簡単なバッチタスクの例として、価格自動変更を行うプログラムを作成することを通し、バッチタスクの基本的な作り方を学んでいきます。このタスクは、次のような仕様とします。

- 商品マスタの全レコードに対し、自動で(ユーザの手作業での介入なしで)価格を変更する。
- 変更する割合（%値）は、パラメータとして渡される。

前章で作成した価格変更プログラムでは、手動価格変更しか実装していませんでしたが、残りの自動価格変更は、本章で作成するプログラムを「自動価格更新」イベントハンドラから呼び出すことにより実装されます。

バッチタスクと言っても、何も難しいことはありません。ユーザとの対話処理がないだけ、単純で簡単だともいえます。

13.1. 価格自動変更バッチタスクの作成

13.1.1. タスク特性

まず、自動変更タスクのタスク特性を定義しましょう。
自動変更タスクは、商品マスタの全レコードを対象として処理を行いますので、メインテーブルは商品マスタとなります。
タスクタイプは、ユーザとの対話的操作がないので、バッチタスクとします。
初期モードは、データの更新を伴う処理であるため、修正モードとします。
以下のようにして、タスク特性を設定してください。

1. プログラムリポジトリの最下行に新しいタスクを作成し、開きます。
2. タスク特性で次のようにパラメータを設定します。

名前	価格変更バッチ
タスクタイプ	B=バッチ
初期モード	M=修正
メインテーブル	3 商品テーブル
	1 商品番号

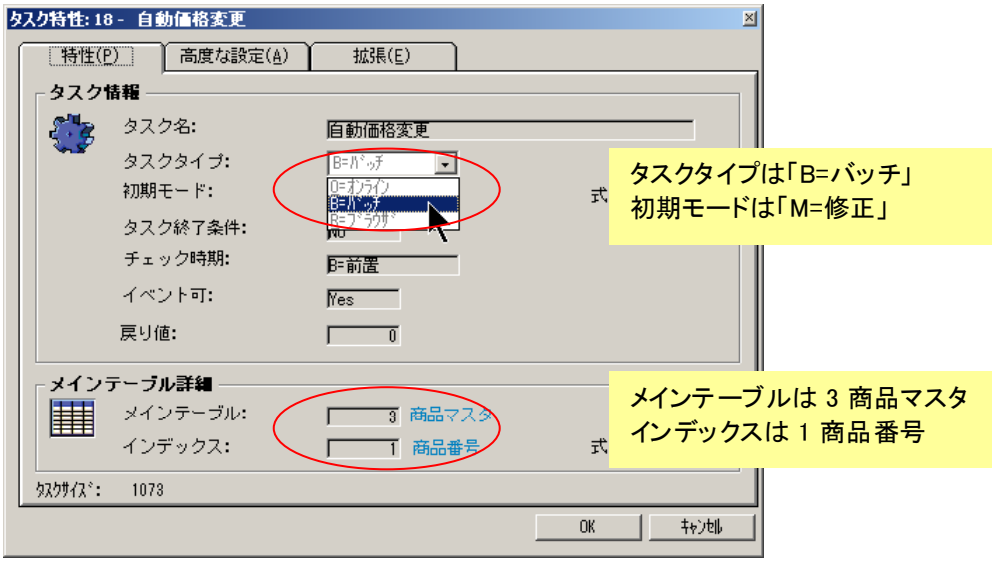


図 13-1 バッチタスクのタスク特性

13.1.2. レコードメイン

次に、タスクで必要とされるデータ項目をレコードメインで登録します。
自動更新タスクで必要とされるデータ項目としては、

- 変更%の値を受け取るパラメータ項目
- メインテーブルのキーとなる商品番号カラム（実データ項目）
- 更新の対象となる単価カラム（実データ項目）

があります。
次のようにして、レコードメインでセレクトコマンドを使い、これらのデータ項目を登録してください。

1. レコードメインを開きます。
2. 変更%を受け取るパラメータ項目を定義します。
   ① 1行作成して、セレクト P=パラメータ とします。
   ② 番号欄からズームすると、パラメータテーブルが開きます。
   ③ モデルで、「その他%」を選択します。
   ④ 名前を「変更%」とします。
   ⑤ パラメータテーブルを閉じます。
3. 「セレクト R=実データ」により、「商品テーブル」から、「1 商品番号」と「4 単価」の二つのカラムを選択します。

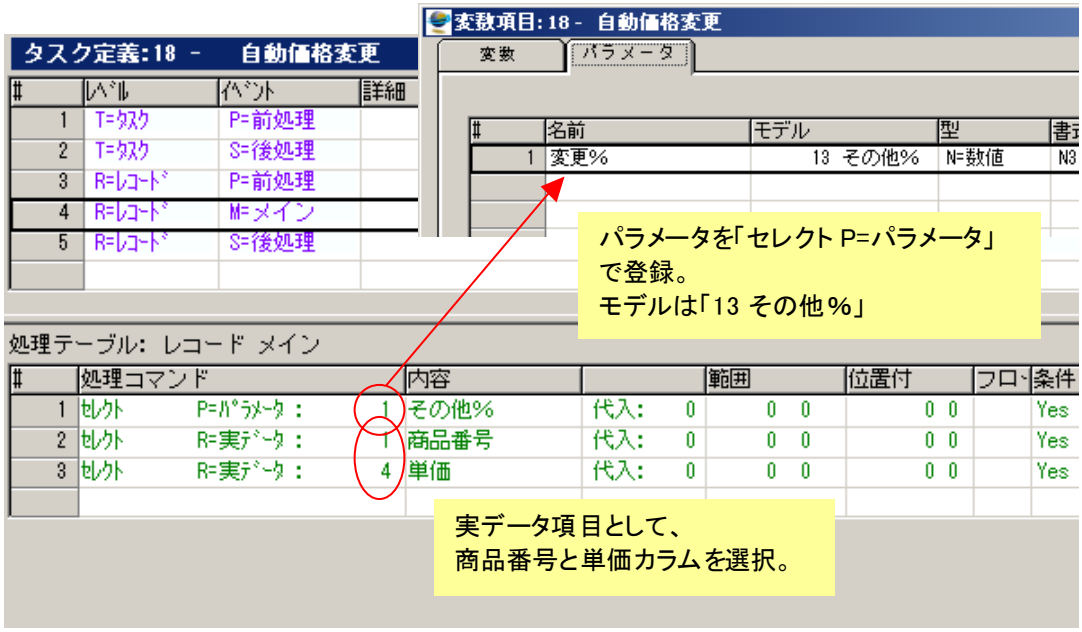


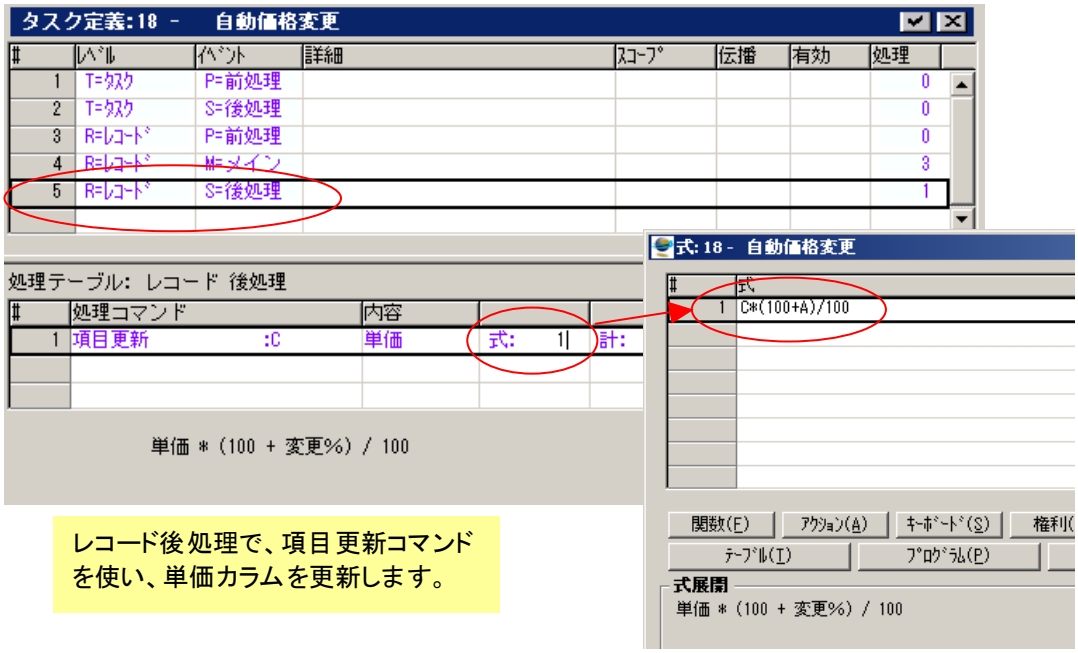
図 13-2 パラメータ「変更%」と、実データ項目「商品番号」と「単価」

13.1.3. レコード後処理

対象となるレコードのすべてに処理を行うレベルはレコード後処理であるので、レコード後処理で、項目更新コマンドを使って、単価の更新を行います。

1. レコード後処理を開きます。
2. 1行作成します。
3. 次のような項目更新コマンドで、単価カラムを更新します。

項目更新 C(単価) 式： 単価 * (100 + 変更%) / 100



レコード後処理で、項目更新コマンドを使い、単価カラムを更新します。

図 13-3 レコード後処理で、単価を更新

以上で自動価格変更プログラムは出来上がりです。

13.2. 価格変更プログラムから呼び出す

前節で作成した自動価格変更プログラムは、変更%を指定して、別プログラムから呼び出して実行します。ここでは、前章で作成した「価格変更(2)」プログラムの、「自動価格更新」イベントハンドラから呼び出すようにします。

13.2.1. 変更%のための変数項目を作成

「価格変更(2)」プログラムには、変更%をユーザが入力できる項目がありませんので、ここで追加しておきます。

1. プログラムリポジトリから、前章で作成した「価格変更(2)」プログラムを開きます。
2. レコードメインを開きます。
3. 「手動価格変更」と「自動価格変更」の二つのセレクトコマンドの間に、新たにセレクトコマンドを作成します。(手動価格変更のセレクトコマンドの上にカーソルがある状態で、F4 キーを押すと、1 行新規に作成されます)。

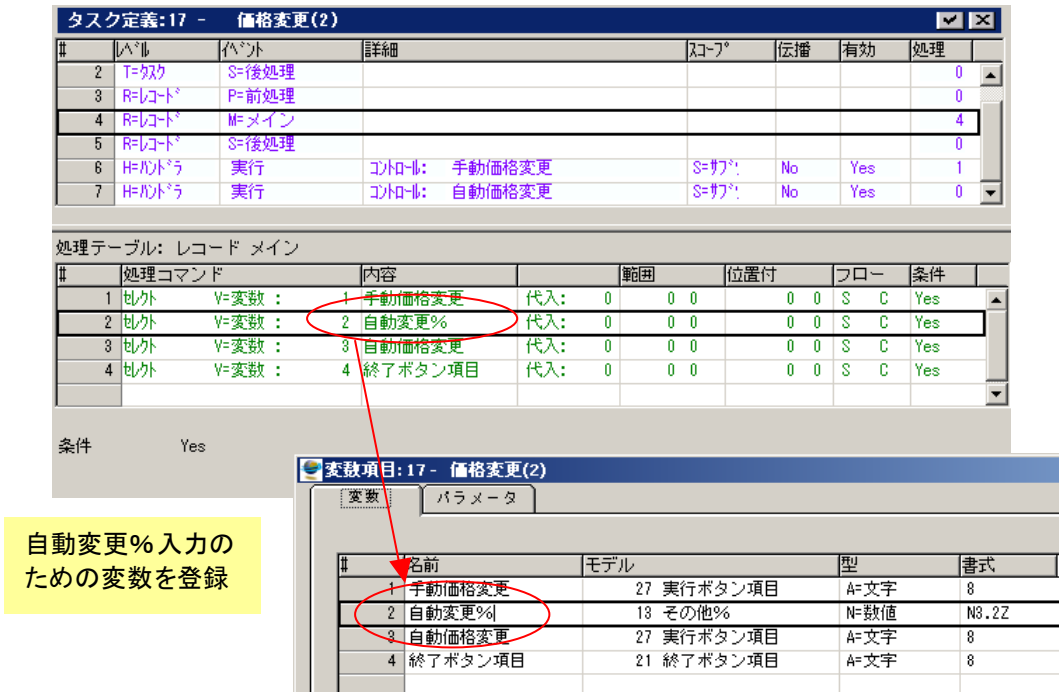


図 13-4 価格変更(2)プログラムに、自動変更%入力のための変数を登録

13.2.2. フォームの修正

上で作った自動変更%をフォームに配置します（図 13-5)。これに伴い、ボタンの位置やフォームのサイズなども適当に変更してください。

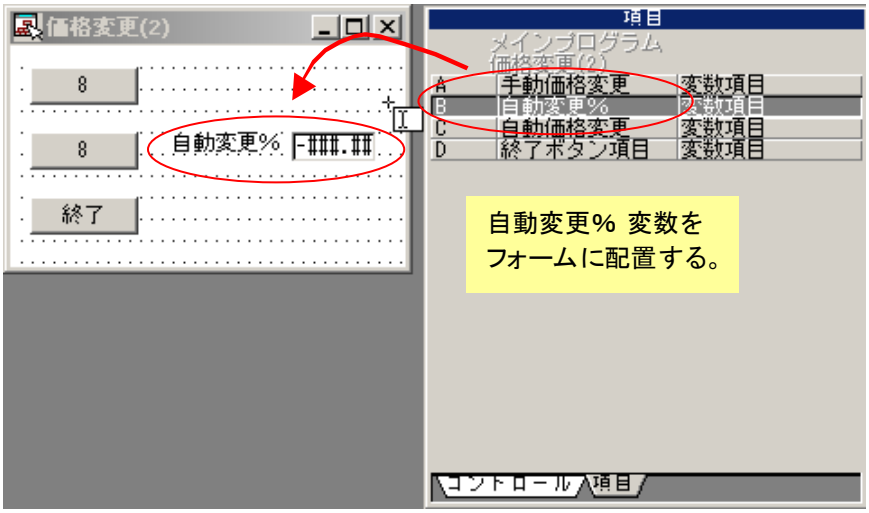


図 13-5 自動変更%の項目をフォームに配置

13.2.3. イベントハンドラからコール

これで準備が整いましたので、空のままになっていた「自動価格変更」のイベントハンドラに、コールコマンドで「自動価格変更」のバッチプログラムを呼び出すようにしましょう。

参考： パラメータとして「自動価格%」を渡す必要がありますが、これはコールコマンドの「パラ：」欄からズームしてパラメータテーブルを開き、指定することができます。

1. ハンドラが「H=ハンドラ」、イベントが「実行」、コントロールが「自動価格変更」であるハンドラを開きます。
2. 処理テーブルで 1 行作成し、次のようなコールコマンドを登録しあす。

   コール P= プログラム 18 自動価格変更
3. 「パラ：」欄に移り、ズームします。
4. パラメータテーブルが開くので、F4 で新規行を作成し、「項目」に B (自動変更%) を指定します。
5. パラメータテーブル、タスクを閉じます。

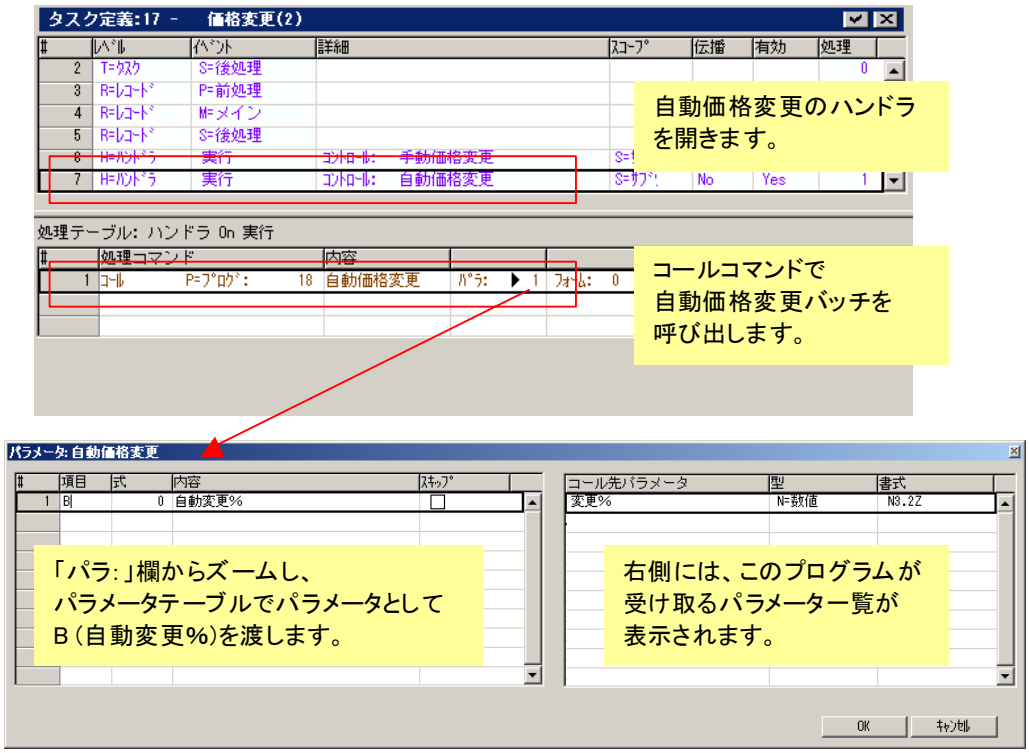


図 13-6 自動価格変更のイベントハンドラから、バッチプログラムを呼び出し

13.2.4. 実行してみる

これでできたので、実行して動作を確認してみましょう。

1. 最初に、「商品マスタ更新」プログラムを実行して、いくつかの商品の価格を記録しておきます。例えば、1002 プードルは 10,710 円です。
2. 「価格変更(2)」プログラムを F7 で起動します。
3. 「価格変更%」に 5 を入力します。つまり 5%の値上げをします。
4. 「自動」ボタンを押します。レコード数が少ないので、処理はすぐに終了します。
5. 「終了」ボタンを押して、プログラムを閉じます。
6. 再度、「商品マスタ更新」プログラムを実行し、商品の価格を実行前と比較してみてください。プードルは 5%の値上がりで 11,246 円となっているはずです。

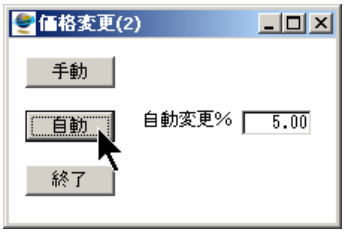


図 13-7 自動価格変更の実行

14. メニュー

今までは、プログラムをプログラムリポジトリから F7 キーを使って実行してきました。
実際のシステム運用では、エンドユーザがそのような操作をすることはなく、通常は図 14-1 のようなメニューを使ってプログラムを実行させます。

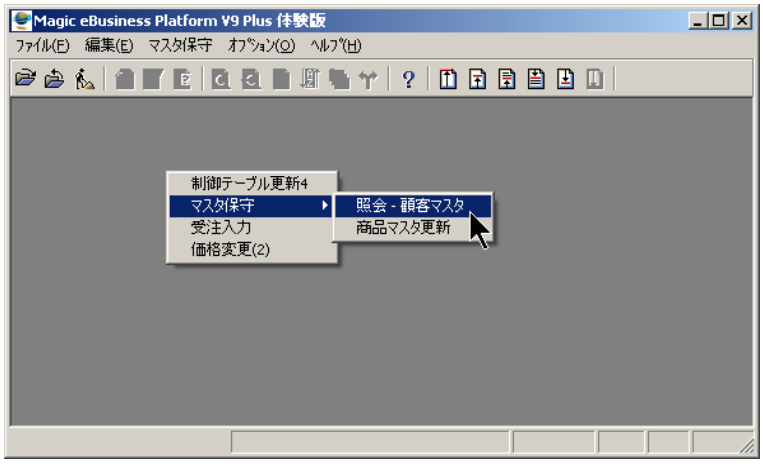


図 14-1 実行時メニュー（コンテキストメニュー）

Magic は、2 種類のメニューを用意しています。図 14-1 に示したものは、マウスの右ボタンクリックで表示されるメニューで、**コンテキストメニュー**と呼びます。
もう 1 種類のメニューは、図 14-2 に示すようにウィンドウのメニューバーに登録するもので、**プルダウンメニュー**と呼びます。

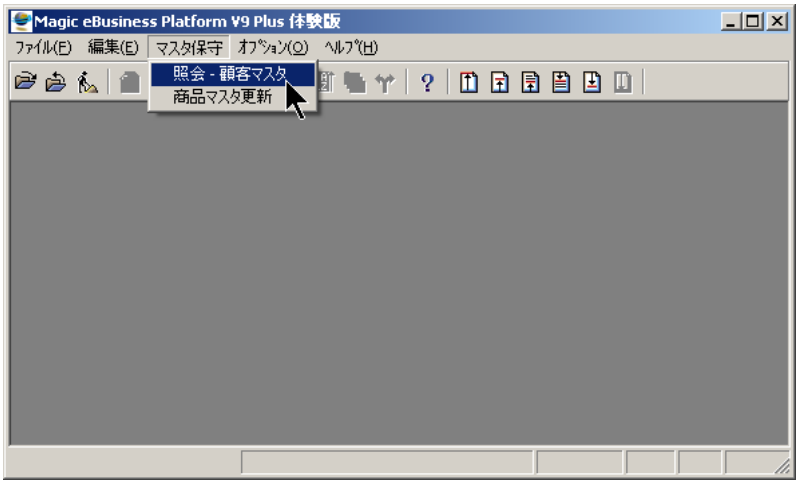


図 14-2 実行時メニュー（プルダウンメニュー）

本章では、Magic の持つメニュー登録機能について説明します。

参考情報： 本章の内容については、リファレンスマニュアル 第 12 章 エンドユーザメニュー&ヘルプ により詳しい情報がありますので、参照してください。

14.1. メニューリポジトリを開く

メニューはすべて**メニューリポジトリ**に登録します。メニューリポジトリは、以下のいずれかにより開くことができます（図 14-3）。

- メニュー 「ワークスペース(W)」から「メニュー」を選ぶ。
- ツールバーから、「メニュー」アイコンをクリックする。
- Shift + F6 キーを押す。

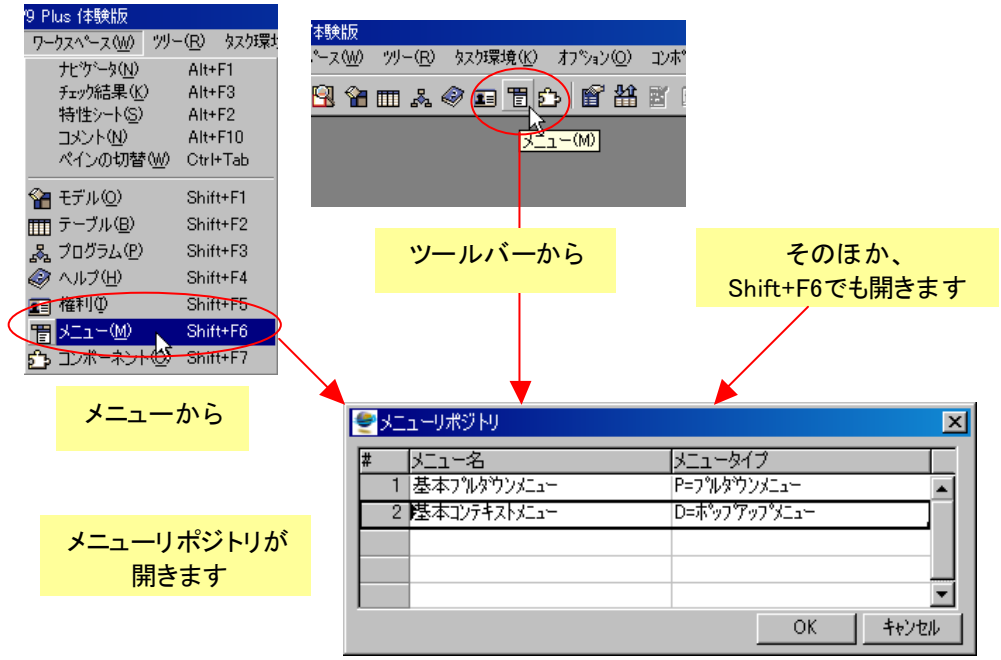


図 14-3 メニューリポジトリの開き方

初期状態では、「基本プルダウンメニュー」と「基本コンテキストメニュー」の二つの行があります。「基本プルダウンメニュー」からズームすると実行時プルダウンメニューを登録する画面が出て、「基本コンテキストメニュー」からズームすると実行時コンテキストメニューを登録する画面が出ます。

14.2. コンテキストメニュー

最初に、コンテキストメニューを登録します。

コンテキストメニューを登録するには、まず、メニューリポジトリの「基本コンテキストメニュー」からズームして、メニュー定義画面を開きます。

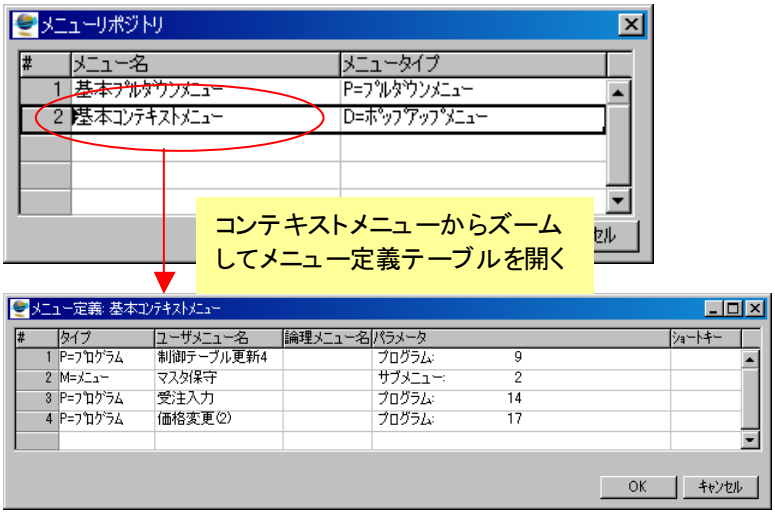


図 14-4 コンテキストメニュー定義テーブルを開く

14.2.1. プログラムを登録

例として、プログラム９番（制御テーブル更新４)をメニューに登録することを考えて見ます。

メニューにプログラムを登録するには、メニュー定義テーブルで、次のように行います。

1. F4 キーで新規行を作成します。
2. タイプは「P=プログラム」とします。
3. パラメータ欄 「プログラム：」に移動しズームすると、プログラム一覧が表示されます。
4. プログラムの中から、呼び出ししたいプログラム（９番 制御テーブル更新４）を選択します。

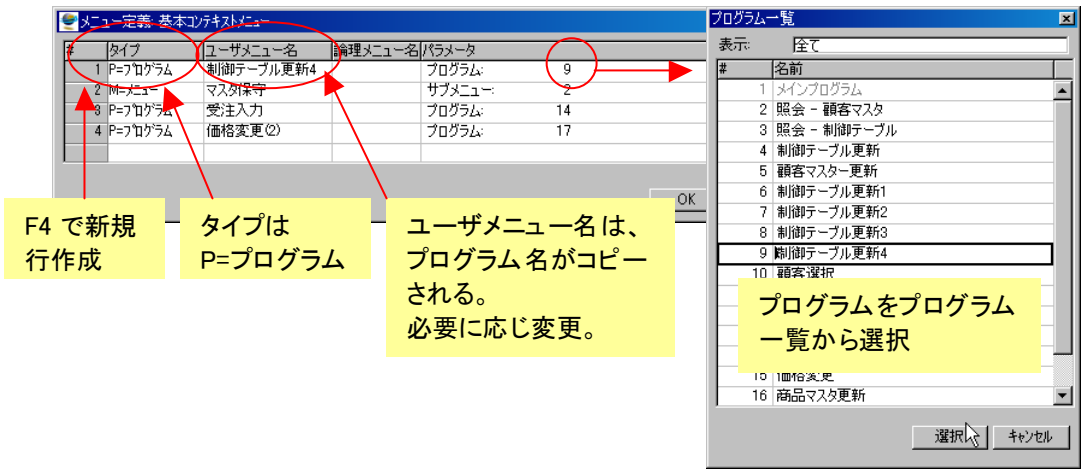


図 14-5 メニューにプログラムを登録

14.2.2. サブメニューの登録

メニューではあるメニュー項目を選ぶと、別のメニューが出てくるようにさせることができます。これを**サブメニュー**と呼びます。

サブメニューは、次のようにして登録します。

1. F4 で新規行を作成します。
2. タイプを「M=メニュー」とします。
3. ユーザメニュー名には、適当な名前(例えば「マスタ保守」)を入れます。
4. サブメニュー： 欄からズームすると、サブメニューのメニュー定義テーブルが開きます。
5. 同様の手順で、サブメニューにプログラムなどを登録します。

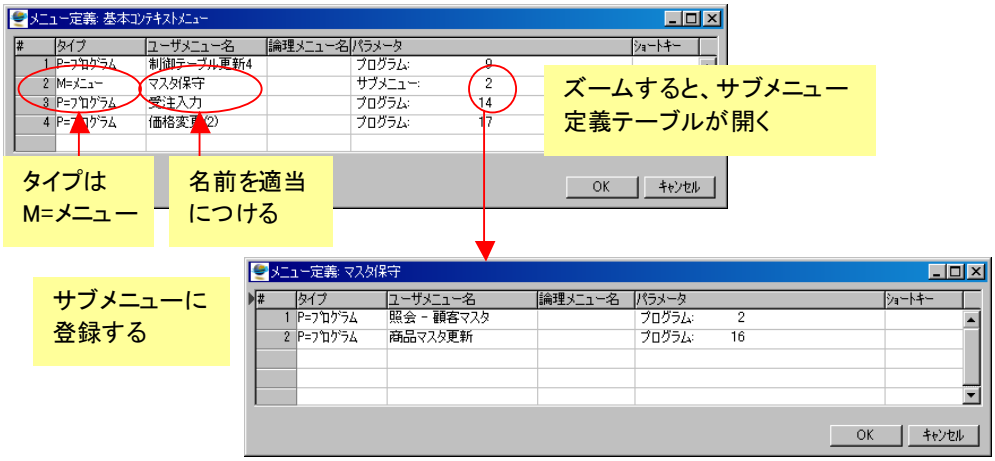


図 14-6 サブメニューの登録

14.3. プルダウンメニュー

プルダウンメニューの登録は、メニューリポジトリで「基本プルダウンメニュー」の行からズームすることを除いては、コンテキストメニューの場合と全く同じです。サブメニューを登録することもできます。

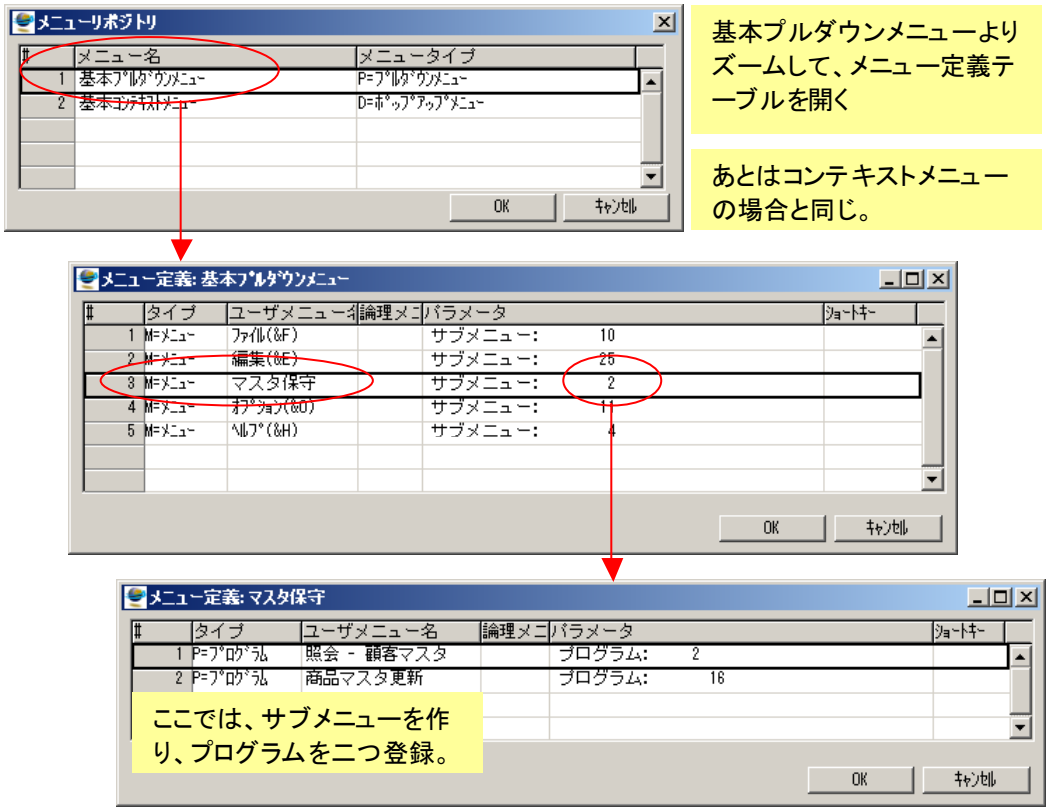


図 14-7 プルダウンメニューの登録

参考： プルダウンメニューには、「ファイル」「編集」などの基本的なメニューがデフォルトでひととおり登録されています。必要なければ適当に削除しても構いません。

以上で、メニュー登録は終わりです。
メニューリポジトリを閉じ、開発モードから実行モードに変えてください。Ctrl + T で変わります。
プルダウンメニューやコンテキストメニューが、図 14-1 や図 14-2 のように、メニューリポジトリで登録されている通りに表示され、プログラムが実行されることを確認してください。

15. ファイル入出力

Magic 以外のプログラムとデータを交換する場合とか、データをテキスト形式で保存したい場合、簡単なレポートを作成する場合などは、テキストデータの入出力の処理が必要となります。
本章では、Magic におけるファイル入出力の基本として、顧客マスタテーブル(Pervasive.SQL 形式のデータ)のデータをテキストファイルの形式に出力する方法、および、テキストファイルデータを読み込んで顧客マスタテーブルに入力する方法について見ていきます。

図 15-1 顧客マスタのテキストファイルデータ（一部）

参考情報： 本章の内容については、リファレンスマニュアルに詳しい情報がありますので参照してください。	
入出力ファイルテーブル	第 6 章 プログラム → 入出力ファイルテーブル
データ出力コマンド	第 7 章 処理コマンド → データ出力
データ入力コマンド	第 7 章 処理コマンド → データ入力
テキスト形式フォーム	第 6 章 プログラム → フォームテーブル
	第 10 章 → 出力フォーム → 入出力フォーム
EOF 関数	第 8 章 関数 → 関数の説明(アルファベット順) → EOF

15.1. ファイル出力

最初に、Magic の顧客マスタテーブルに格納されているデータをすべて、「顧客.txt」という名前のテキストファイルに出力するプログラムを作成します。

15.1.1. APG により作成

このような単純なテキストファイル出力は、APG によりプログラムを作成すれば簡単にできます。

1. プログラムリポジトリを開き、プログラムを一つ追加します。
2. メニュー 「オプション(O)」から「APG(G)」(あるいは Ctrl + G)を選び、APG を起動します。

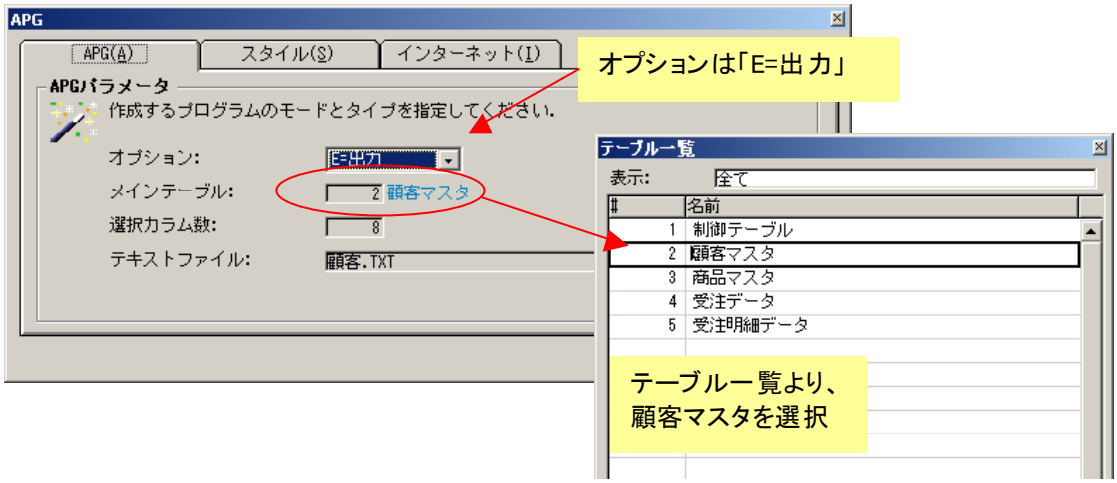


図 15-2 データ出力プログラムを作成する APG

プログラムリポジトリから、 F7 キーにより実行してください。プログラムは終了し、Magic のディレクトリに「顧客.TXT」というテキストファイルができているはずです。中身はテキストデータですので、テキストエディッタで開いて確認してください。図 15-1 のようなデータが格納されているはずです。

15.1.2. APG で作ったデータ出力プログラムを見てみる

では、どのようなプログラムが作られたのか、プログラムを開いて見てみましょう。

タスク特性

タスク特性（図 15-3）で作られたタスクの特性を見ていきましょう。

- タスクタイプはバッチタスクです。ファイル出力は、ユーザとのインタラクティブなやりとりがないタスクなので、バッチになっています。
- 初期モードは Q=照会です。顧客マスタテーブルに対する修正は行わず、読み取り専用となるので、Q=照会モードで実行します。
- メインテーブルは、2 番(顧客マスタ）で、インデックスはデフォルトの 1 です。

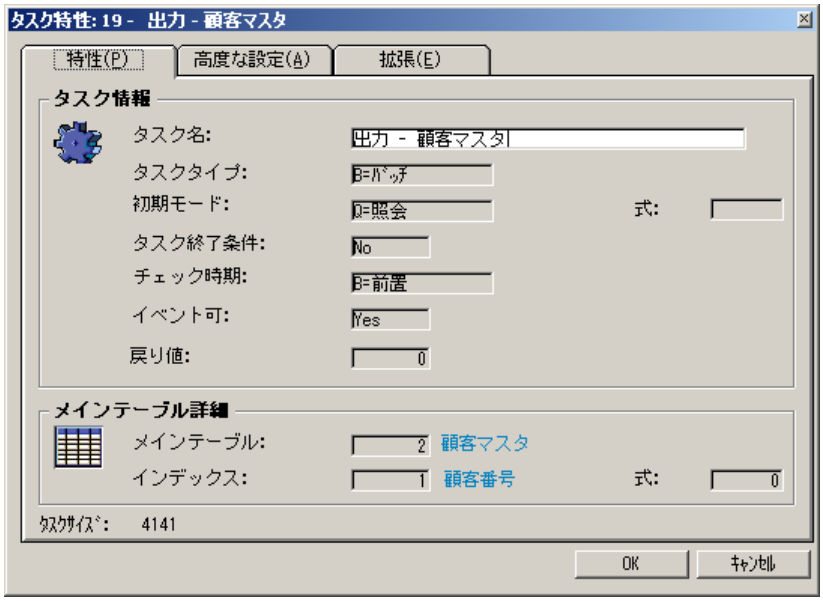


図 15-3 APG で作成した出力プログラムのタスク特性

レコードメイン

レコードメインでは、データ出力の対象となる項目（今の場合には、顧客マスターテーブルの全項目)がセレクトコマンドで定義されています。

図 15-4 レコードメイン

入出力ファイルテーブル

今作っているような、外部のテキストファイルとのファイル入出力があるタスクでは、各ファイルごとに、**入出力ファイル** を登録しておく必要があります。これは、**入出力ファイルテーブル**で行います(図 15-5)。

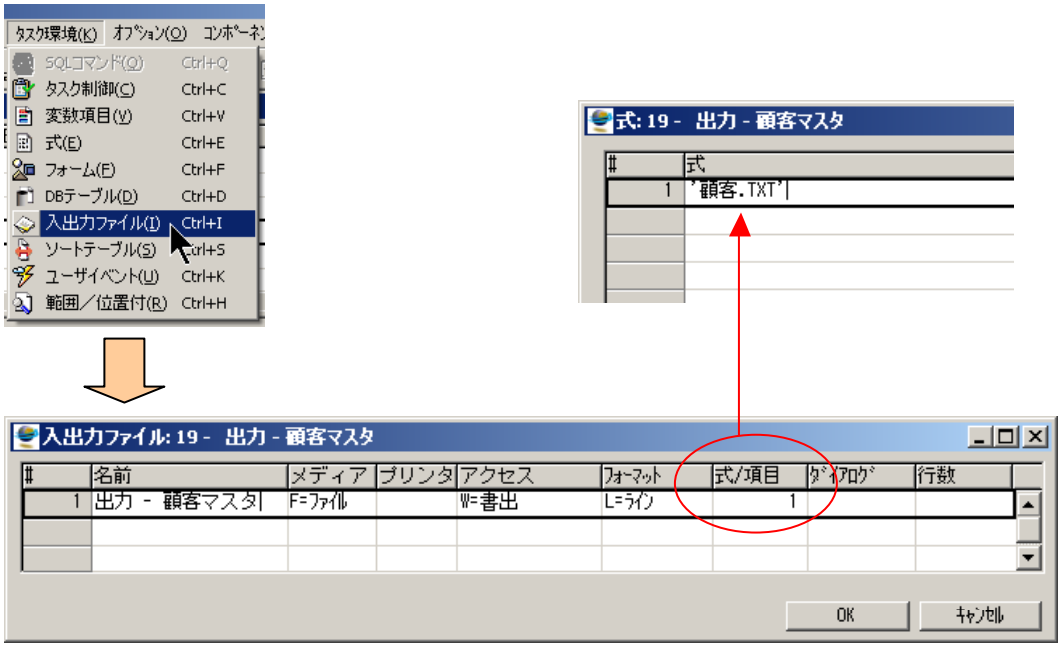


図 15-5 入出力ファイルテーブル

ここでは、データを出力するテキストファイルについて、次のような定義を行います。

- **名前**： Magic のプログラムの中でだけ使う名前です。任意の文字を使うことができます。
- **メディア**： 今回はテキストファイルなので、F=ファイルが選ばれています。メディアとしてはこの他に印刷用にテキスト形式プリンタ、GUI 形式プリンタ、コンソールがあり、その他高度な用途のために、項目、リクエスタ、その他がありますが、本チュートリアルでは扱いません。
- **アクセス**： アクセスの方向を指定します。ファイルにデータを書き込むので、W=書出 が選択されています。
- **フォーマット**： ファイルの場合には、「L=ライン」(1 行ごとに改行コードを入れる)か「N=なし」(改行コードを入れない)が選べます。通常は L=ラインを使います。
- **式/項目**： ファイル名を式で指定します。ズームすると式テーブルが出てきますが、ここで文字列の定数で'顧客.TXT' が定義されています。今回は固定の文字列を使っていますが、ここには任意の文字型の式を使うことができ、パラメータ項目を使って出力ファイル名を渡すとか、ファイル名に日時を追加するというような使い方を良くします。

フォームテーブル

出力するファイルのデータの形式は、フォームによって定義します。

1. フォームテーブルを開いてください。メニュー「タスク環境（K)」から「フォーム(F)」を選ぶか、あるいは Ctrl + F で開きます。
2. 第 3 番目のフォームを選び、ズームしてください。図 15-6 のように、細長いテキスト形式フォームが表示されます。

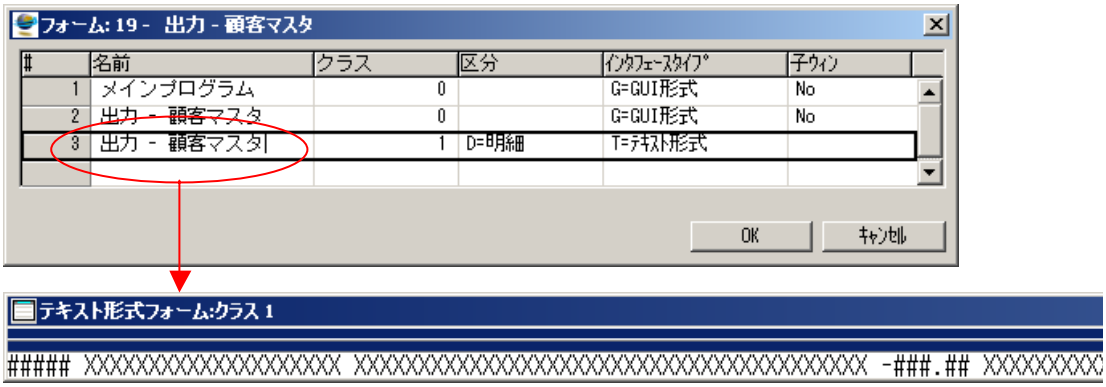


図 15-6 テキスト出力用フォーム

このフォームは、テキストデータ出力用のフォームです。このフォームには全項目が横一列に並んで配置されています。このため縦が 1 行、横が 317 文字の細長いフォームになっていますが、この形式でデータが出力されるので、1 レコードに 1 行づつのテキストデータとなります。

フォームテーブルに定義されている設定について、簡単に解説します：

- **名前：** 名前は、Magic 内部でだけ使う、フォームの名前です。上の例のように同じ名前のフォームが複数あっても構いません。内部では番号でフォームを区別します。
- **クラス：** 0 から 255 までの数字を指定します。0 はユーザにデータを表示したり、ユーザからの入力を受け付ける GUI 形式のフォームとして予約されています。テキストファイルとの入出力を行うには、1 以上の番号を指定します。この数字は、フォームがたくさんある場合にフォームを分類するために使います。
- **区分：** これはページ印刷でヘッダやフッタを使う場合に利用しますが、今回のように 1 レコード 1 行でベタに出力する場合には、D=明細 を指定します。
- **インターフェースタイプ：** フォームとして出力するメディアに合ったフォームのタイプを指定します。今回の場合には単純なテキスト出力ですので、T=テキスト形式を使います。その他のタイプとしては、GUI 形式(グラフィック印刷の場合)、HTML 形式 や HTML マージ形式(Web アプリケーションの場合)などがあります。

レコード後処理でのデータ出力コマンド

あとは、このフォームを使って、1 レコードごとにファイルへ出力すればよいことになります。

ファイル出力は、**データ出力**コマンドにより行います（図 15-7)。1 レコードごとに実行するので、レコード後処理で実行させます。

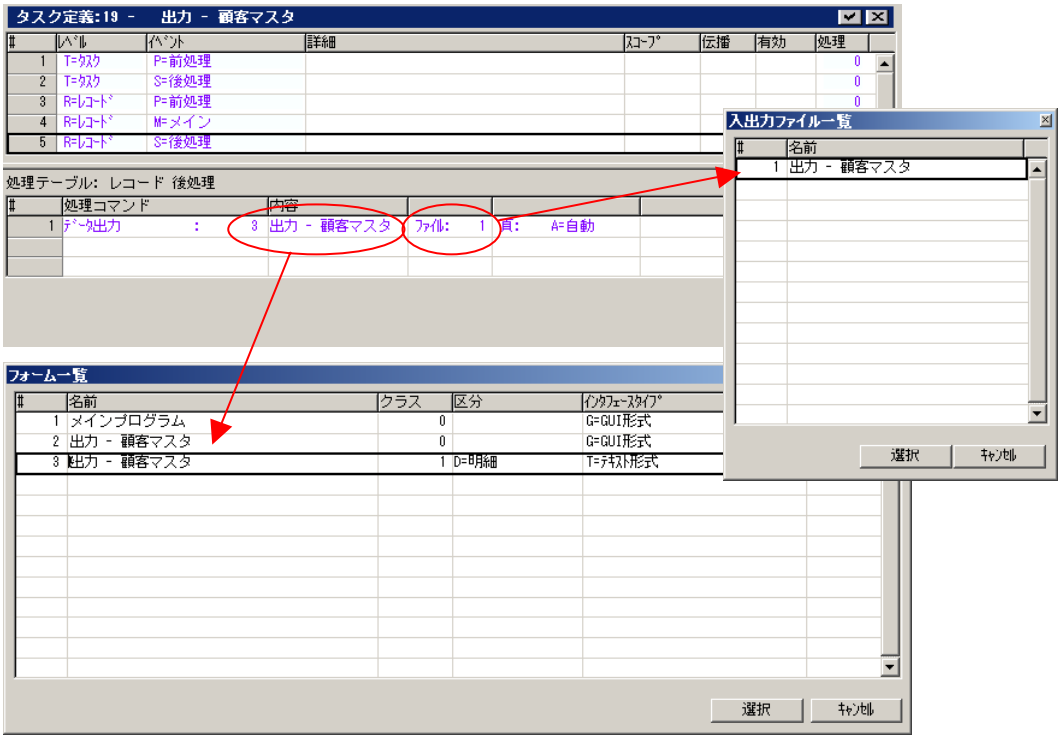


図 15-7 レコード後処理のデータ出力コマンド

データ出力コマンドには、3 つのパラメータがあります。

- **フォーム番号：** データ出力で使うフォームの番号を指定します。ここからズームすると、フォームの一覧が表示されるので、一覧から選択できます。
- **ファイル：** 出力先のファイルを指定します。ここからズームすると、入出力ファイルテーブルに登録されているファイルの一覧が表示されるので、これから選択できます。
- **頁：** ページ形式で印刷する場合に、1 ページに収まる行数に達した場合にどうするかを指定します。今回の場合には 1 行 1 レコードのベタの形式ですのでこの設定は無視されますが、デフォルトの A=自動となっています。

15.1.3. APG を使わずに出力プログラムを作成する

練習のために、同じプログラムを APG を使わずに作ってみてください。

参考： テキスト出力用フォームを作成する場合、フォームの APG を使うと便利です。フォームの APG は、レコードメインで定義されているデータ項目すべてを自動的にフォームに配置する機能で、次のようにして実行できます。

1. フォームテーブルを開きます。
2. フォームを一つ追加します (既存のものを上書きすることも可能です)。
3. クラスは 1、インターフェースタイプは T=テキスト形式フォーム とします。
4. メニュー「オプション(O)」から「フォーム自動作成(F)」を選びます。
5. 「作成フォーム」ダイアログが出てきます。今回はテキスト形式フォームで、ラインモード(1 行に 1 レコードの形式)で出力するので、デフォルトのままとします。
6. OK ボタンを押すと、フォームが自動作成されます。
7. ズームして、作成されたフォームを確認してください。

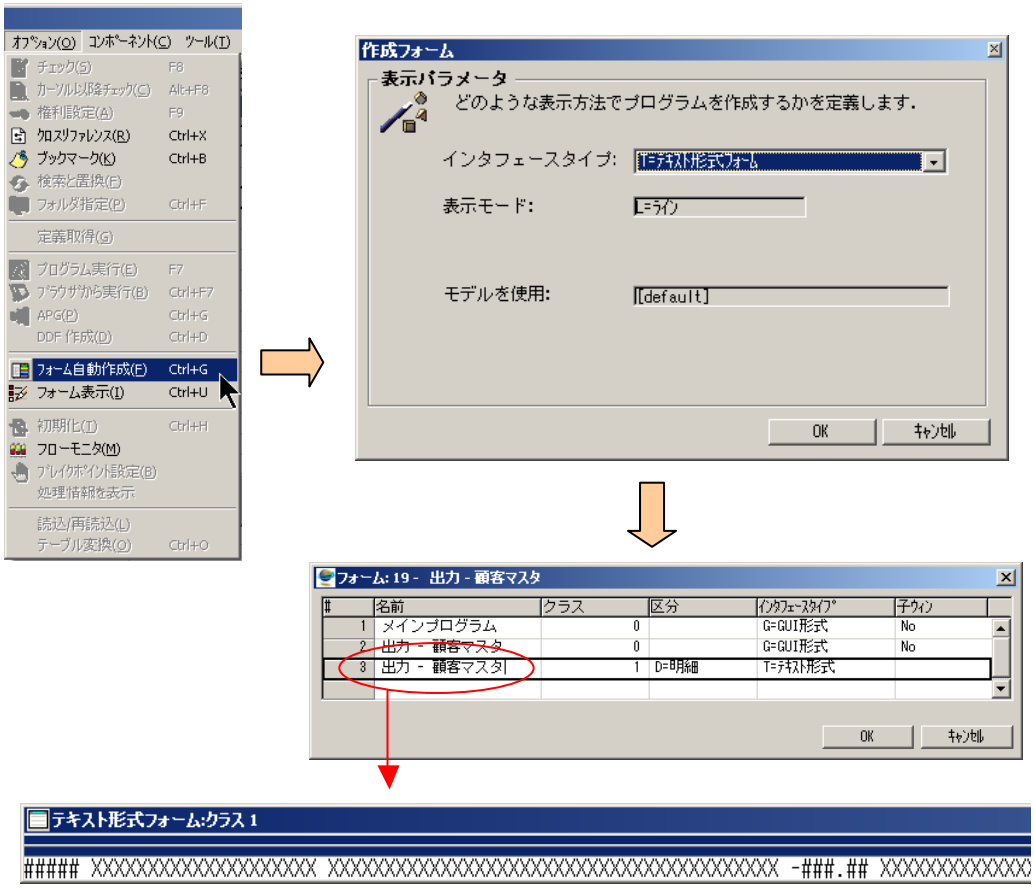


図 15-8 フォーム APG

15.2. ファイル入力

次は、テキストファイルからデータを読み込むプログラムを作成しましょう。

15.2.1. APG により作成

ファイル入力についても、APG でプログラムを作成するのが一番です。

1. プログラムリポジトリを開き、新規プログラムを作成します。
2. Ctrl + G で APG を起動します。
3. オプションは I=入力、メインテーブルは 2 (顧客マスタ)とし、後はデフォルトのままとします。
4. OK ボタンを押せば、プログラムが作成されます。

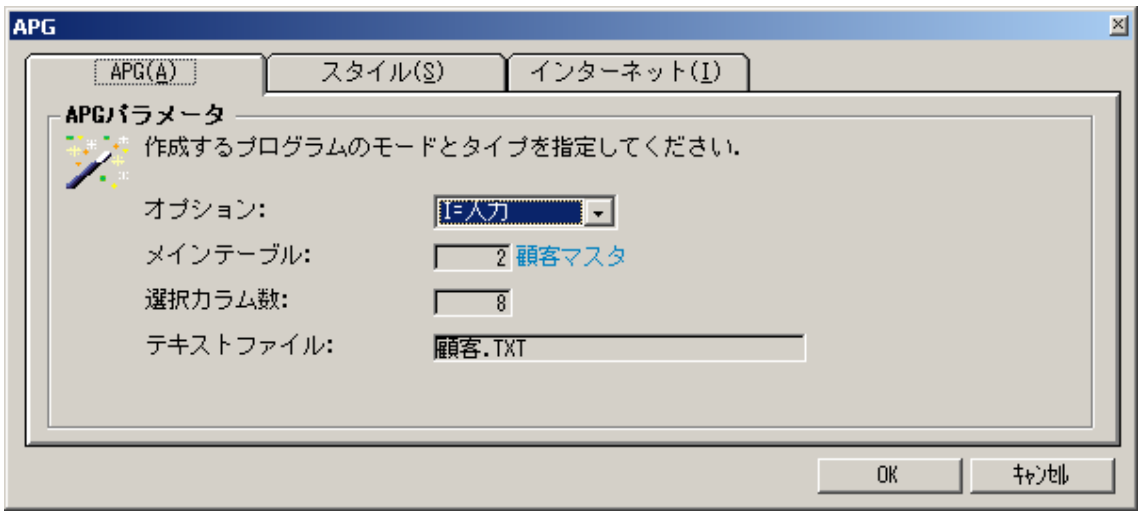


図 15-9 データ入力プログラムを作成する APG

15.2.2. 実行してみる

このプログラムを実行するにあたっては、事前に顧客マスタテーブルを空にしておく必要があります。すでにデータが入っているテーブルに、改めてデータを入力すると、同一顧客番号のデータを登録しようとすることになるので、キー重複のエラーが出てしまうからです。

顧客マスタテーブルを空にする一番簡単な方法は、顧客マスタに対応する Pervasive.SQL のファイル 顧客.MST を削除してしまうことです。ここでは安全のために削除はせずに、別名に変更しておくことにしましょう。エクスプローラなどで Magic のディレクトリを開き、そこにある 顧客.MST ファイルを、顧客_00.MST などにリネームしてください。

この状態で、テーブルリポジトリを開き、顧客マスタテーブルを APG で開いてみてください。レコードが 1 件もない空の状態になっているはずです。

次にテーブルリポジトリに戻り、今 APG で作成した「入力 – 顧客マスタ」を F7 で実行します。すぐに終了するはずです。

再度テーブルリポジトリを開き、顧客マスタテーブルを APG で開いてみてください。今度は顧客のレコードが登録されているはずです。

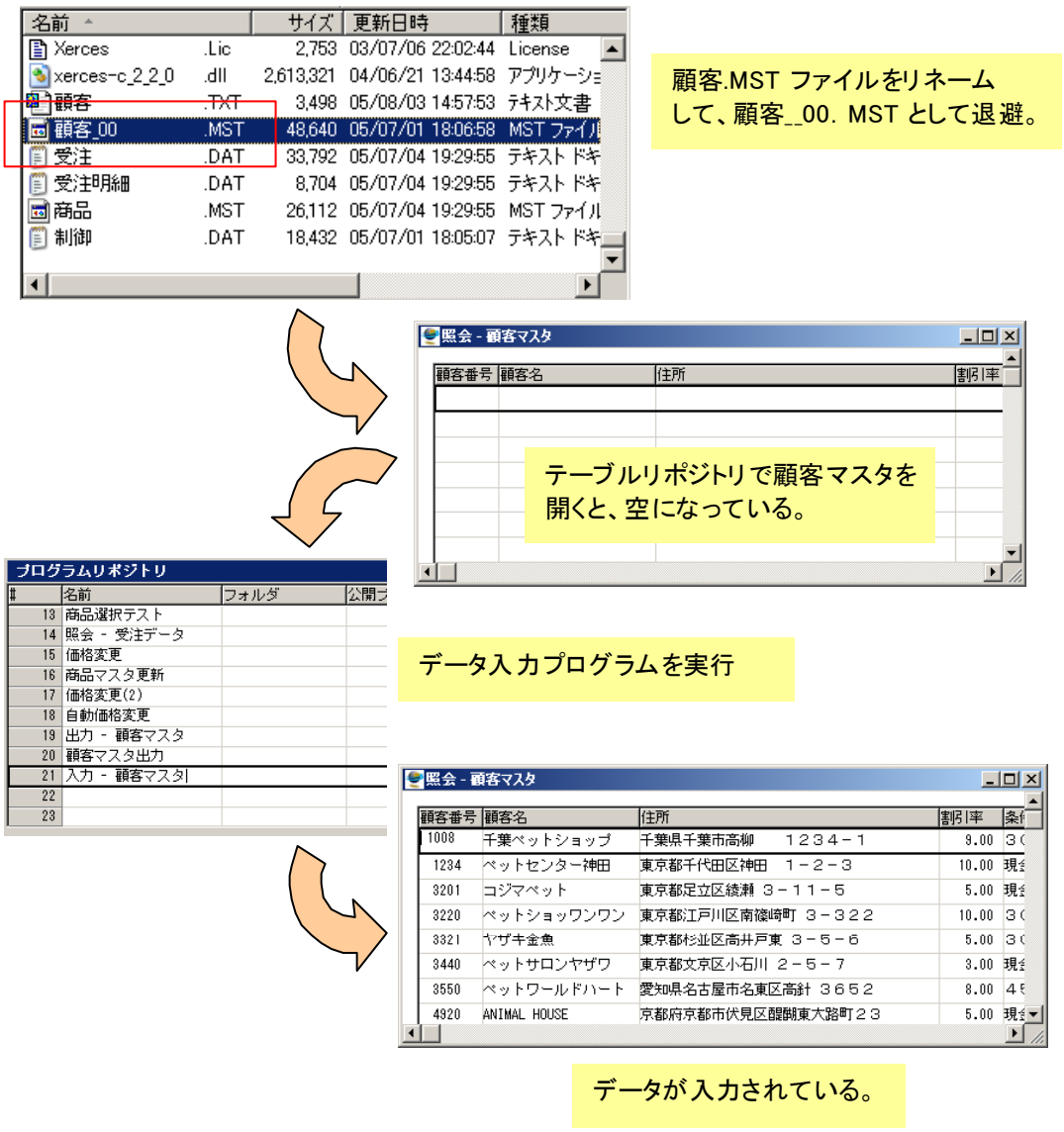


図 15-10 APG で作成したデータ入力プログラムの実行

15.2.3. APG で作ったデータ入力プログラムを見てみる

それでは、今作ったプログラムを開いて、どのようにプログラムが作られているかを見ていきましょう。

タスク特性

タスク特性（図 15-11）は次のようになっています。

- **タスクタイプ：** データ出力の場合とおなじく、ユーザとのインタラクションがないので、バッチタスクになっています。
- **初期モード：** データ入力の場合には、顧客マスタテーブルにレコードを登録していくので、登録モードとなっています。
- **タスク終了条件：** このタスクは、テキストデータを読み込んでレコードを作成していきますが、テキストデータをすべて読み込んだときに終了します。そのため、終了条件は、式で EOF(0,1) と設定されています。

  参考： EOF 関数の第一引数は、タスクの世代を表し、0 は自分を表します。第二引数は入出力ファイル

テーブルに登録されている入出力ファイルの番号を示します。EOF 関数は、第 1 引数で指定された世代のタスクに定義されている、第 2 引数で指定された入出力ファイルが、ファイルの終わりに達しているかをチェックし、まだ達していなければ False を、達していたら True を返します。

- **メインテーブル：** 顧客マスタにデータが入力されるので、顧客マスタのテーブル番号 2 が指定されています。
- **インデックス：** 今の場合、レコードを入力するだけなので、インデックスの指定は意味がありません。デフォルトの 1 が設定されています。

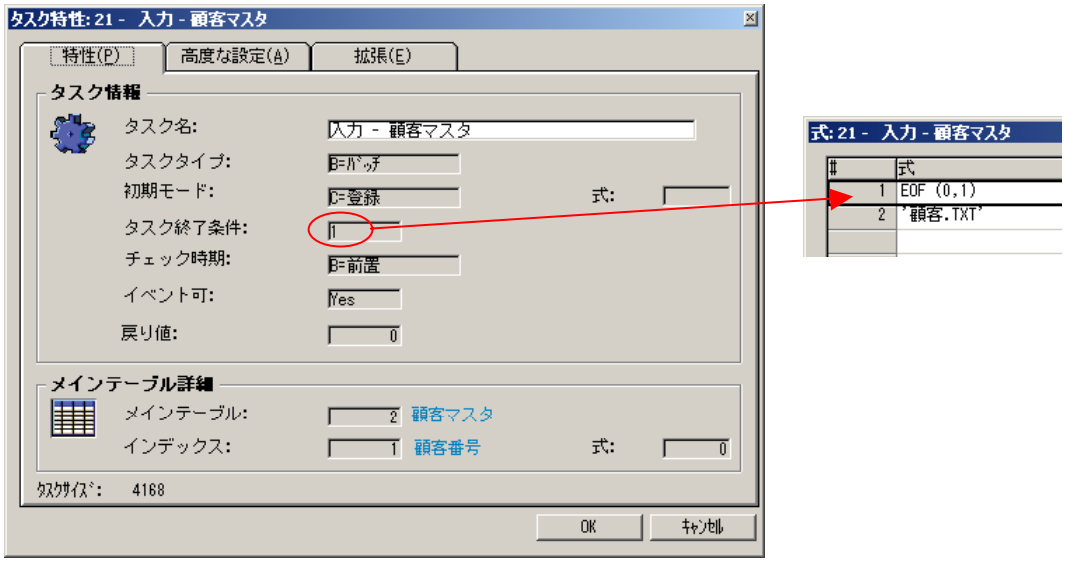


図 15-11 データ入力プログラムのタスク特性と終了条件

レコードメイン

顧客マスタの全項目についてデータを入力してくるので、レコードメインでは全項目をセレクトコマンドで選択しています。

入出力ファイルテーブル

データを読み込んでくるテキストファイルも、入出力ファイルテーブルで定義します(図 15-12)。

入出力ファイルテーブルは、メニュー「タスク環境(K)」から「入出力ファイル(I)」を選ぶか、あるいは Ctrl ＋ I で開くことができます。

出力プログラムの場合と似ていますが、「アクセス」パラメータが、「W=書出」から「R=読込」になっています。「式」欄には、入力テキストファイル名を式で指定します。

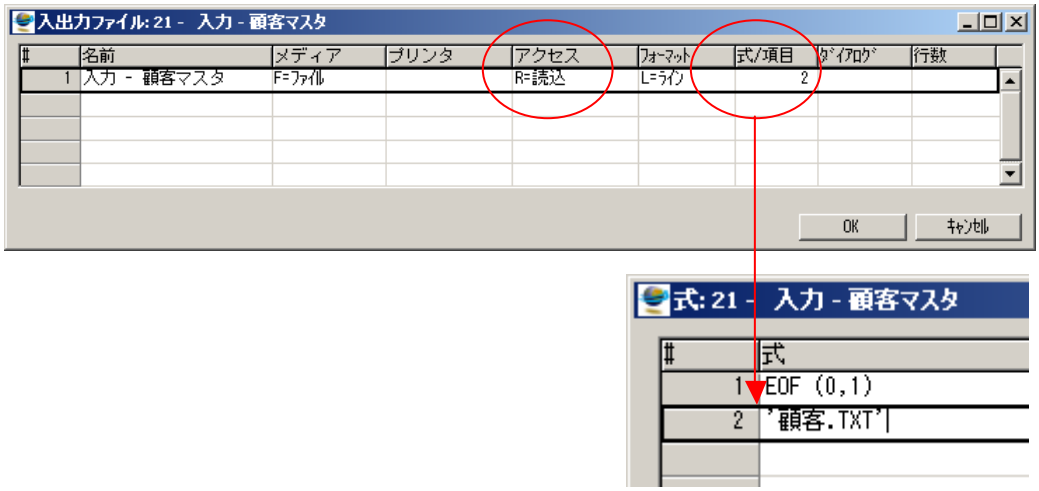


図 15-12 入出力ファイルテーブル

フォーム

テキストファイルからデータを読み込む場合にも、テキスト形式のフォームを用います。

実際にはこのフォームは、データ出力の場合と全く同じです。出力のときと入力のときとで、データの形式が合っていなければならないからです。

フォームテーブルを開いて、データ入力用のフォーム(3 番)をズームし、内容を確認してください。

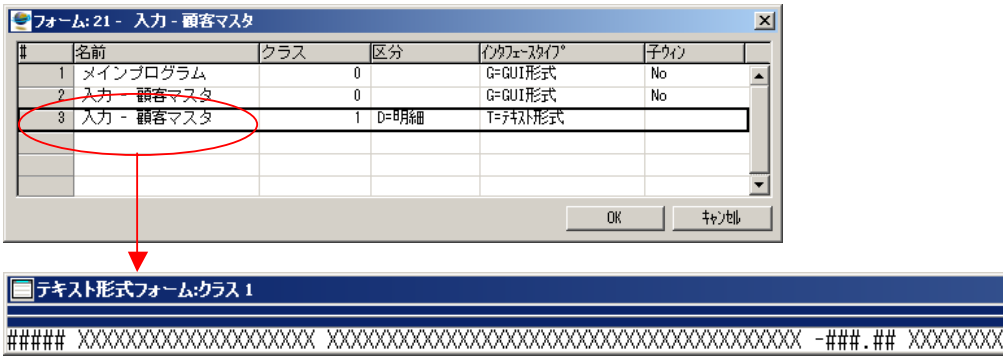


図 15-13 データ入力用フォーム

レコード後処理でのデータ入力コマンド

データ出力の場合と同じく、レコード後処理で、1 レコード分のデータを読み込みます。

データの読み込みは、データ出力コマンドと対になる **データ入力** コマンドで行います。

データ入力コマンドは、データ入力コマンドと同じく、フォームとファイルのパラメータがあります（図 15-14)。

フォームは、データ入力で使うテキスト形式フォームを番号で指定します。この欄からズームすると一覧が表示されるので、一覧から選択することもできます。

ファイルは、データを読み込む入出力ファイルを番号で指定します。この欄からズームすると一覧が表示されるので、一覧から選択することもできます。

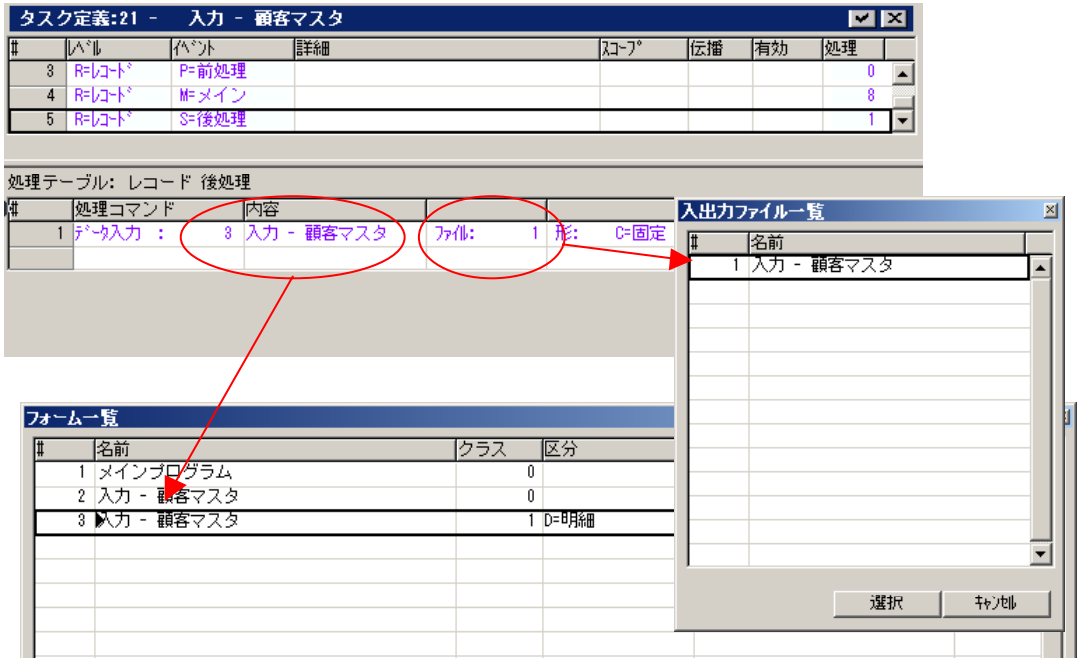


図 15-14 レコード後処理でのデータ入力コマンド

15.2.4. APG を使わずに入力プログラムを作成する

練習のために、同じプログラムを APG を使わずに作ってみてください。

16. マルチユーザ環境

今日の情報システムでは、複数のユーザが同時に同一のリソース(データベース中のレコードなど)をアクセスするのが普通ですが、このような環境では、複数ユーザによる同時アクセスの制御を正しく設計しておかないと、データ更新の不正、ロック待ちなどの問題が起こります。これは Magic を使ったシステムに限らず、どの情報システムでも同じことで、マルチユーザ環境でアクセスを正しくコントロールすることを**並行制御**と呼びます。

Magic では、並行制御のためにさまざまな機能が備わっており、これを上手に使うことにより、わずかなプログラミングの追加でデータ更新の不正を防ぎつつ不要なロック待ちを減らすことができます。

本章では、最初にマルチユーザ環境での一般的な問題 (更新の喪失とレコードロックの問題)について説明し、次に Magic の並行制御機能について説明します。その後、具体的な例として、チュートリアルで作成したペットショップデモを取り上げ、並行制御を考慮せずに作ったプログラムを複数ユーザ環境で実行した場合に起こる問題をとりあげ、どのようにして回避してゆくのかを説明します。

なお、並行制御の問題は、アプリケーションで使っている DBMS がサポートする並行制御の機能に大きく依存する部分がありますが、本章では基本として Pervasive SQL を DBMS として利用することにします。他のリレーショナル型のデータベース (Oracle、MS-SQL Server、DB2UDB など)を利用する場合には、ロックに加えて**トランザクション**の概念が入ってきますが、トランザクションがからんでくると並行制御の問題は複雑になりがちなので、ここでは Pervasive SQL (Btrieve)でトランザクションを使わないでアプリケーションを作成していきます。

参考情報： 本章の内容については、リファレンスマニュアルに詳しい情報がありますので参照してください。

マルチユーザ対応全般	第 23 章 マルチユーザ環境
動作環境設定	第 2 章 設定 → 設定／動作環境 → [マルチユーザ]タブ
ロック方式パラメータ	第 6 章 プログラム → タスク → [タスク特性]ダイアログ → [拡張]タブ
DB テーブル	第 6 章 プログラム → タスク → DB テーブル

16.1. 更新の喪失

マルチユーザ環境での一番根本的な問題は、**更新の喪失**と呼ばれる問題です。マルチユーザ環境では同一のレコードを複数のユーザが同時にアクセスし更新する可能性があるので、あるユーザが更新したレコードを、他のユーザが上書きしてしまって、結果として、最初のユーザが行った更新が失われてしまう可能性があります。これは当然、データの不正となってしまいますので、常に正確な情報を維持しておくべき情報システムからは排除しなければならない問題です。

簡単な例で見てみましょう。これは、同一の商品レコードを、ユーザ A と B とが同時に更新した例です。

最初には、商品番号#1002 の発注数は 0 であったとします。ユーザ A が発注数を +3 し、ユーザ B が発注数を +2 したとすると、最終的な発注数は 0 + 3 + 2 = 5 でなければなりません。

図 16-1 は両者の更新が正しく行われた例です。この図は上から下に時間が流れていく時系列的に描いたもので、次のような順番で処理が進みます。

1. ユーザ A が商品番号#1002 のレコードを読み込みます。このとき、発注数は 0 です。
2. ユーザ A が＋３し、発注数は 3 となります。
3. ユーザ A がレコードを書き込みます。DBMS 中のレコードに、更新が反映されます。
4. 次にユーザ B が同じレコードを読み込みます。発注数は 3 です。
5. ユーザ B が＋２を計算し、発注数は 5 となります。
6. ユーザ B がレコードを書き込みます。DBMS 中のレコードの発注数は 5 となります。

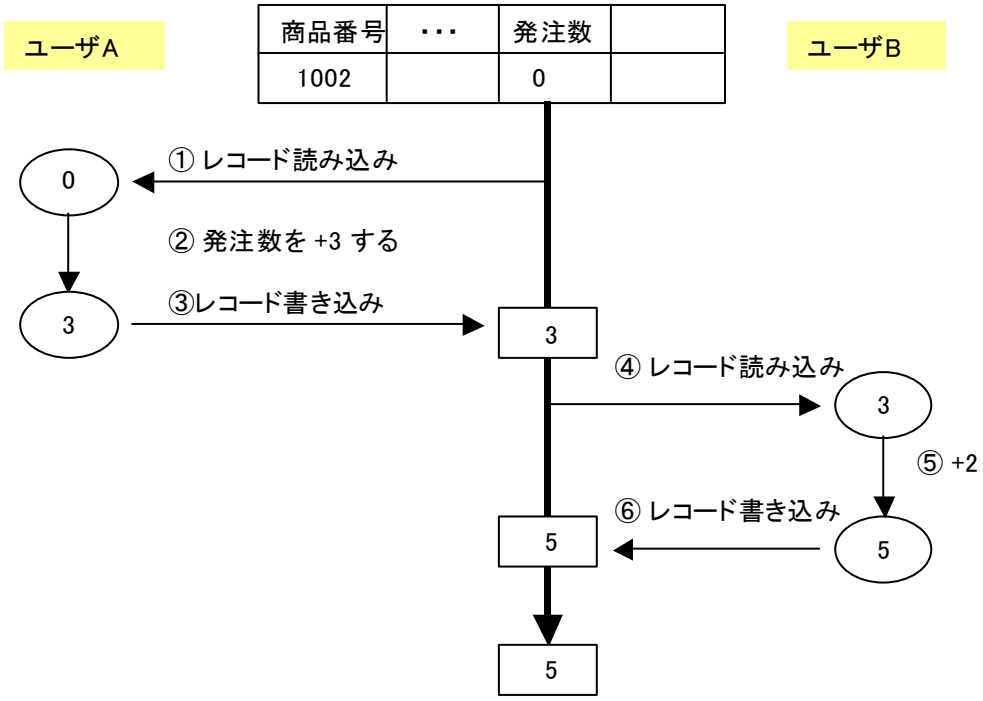


図 16-1 ユーザ A とユーザ B が発注数を更新

ところが、図 16-2 のようなタイミングでレコードが更新された場合には、ユーザ A が行った +3 の更新は、ユーザ B が直後に行った +2 の更新によって、上書きされてしまいます。その結果、最終的な発注数は、5 ではなく 2 になってしまいます。これは明らかに問題です。

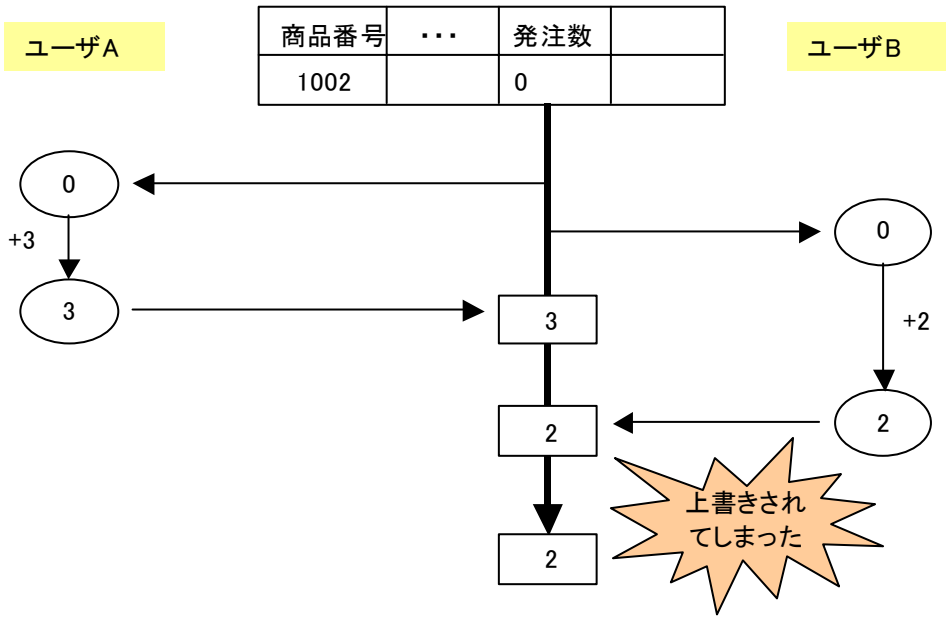


図 16-2 更新の喪失の例

このような問題が起こる原因は、図 16-2 を見ればすぐにわかるように、ユーザ A が結果 13 の書き込みをする前に、ユーザ B がレコードを読み込んでしまい、現在の発注数を 10 として計算してしまうからです。
このように、マルチユーザ環境では、レコードの読み込みと書き込みのタイミングが重要であることがわかります。

16.2. レコードロック

更新の喪失の問題を解決するために、DBMS にロックという機構が導入されました。ロックというのにはいろいろなバリエーションがありますが、基本的な**共有ロック**では、あるレコードに対して、ロックをかけたユーザが独占的な更新の権利を持つことができ、他のユーザはこのレコードにロックをかけたり更新をしたりすることができない、というしくみです。

先ほどの例でロックを使うとどのように更新の喪失を防ぐことができるかを、図 16-3 に示します。

1. 最初にユーザ A がレコードを読み込みます。このとき、レコードの読み込みと同時に、このレコードにロックをかけます。図中では、レコードにロックがかかっている期間を赤色の線で表しています。
2. 次に、ユーザ B が同じレコードを読み込むと同時にロックをかけようと試みます。しかし、このときにはユーザ A がすでにこのレコードにロックをかけてしまっているので、**ロックの衝突**によりユーザ B は失敗します。ユーザ B はこのようなときには、しばらく時間をおいてから再度読み込むようにします。
3. ユーザ A は、＋3 の更新が終わり、レコードに書き込みます。このとき同時に、レコードのロックを解除します。
4. しばらくして、ユーザ B が再度レコード読み込みとロックを試みます。このときにはすでにユーザ A によるレコードロックは解除されているので、読み込みとロックが成功します。
5. ユーザ B は＋2 の更新を行い、レコードを書き込み、ロックを解除します。

結果として、受注数は正しく 5 となります。

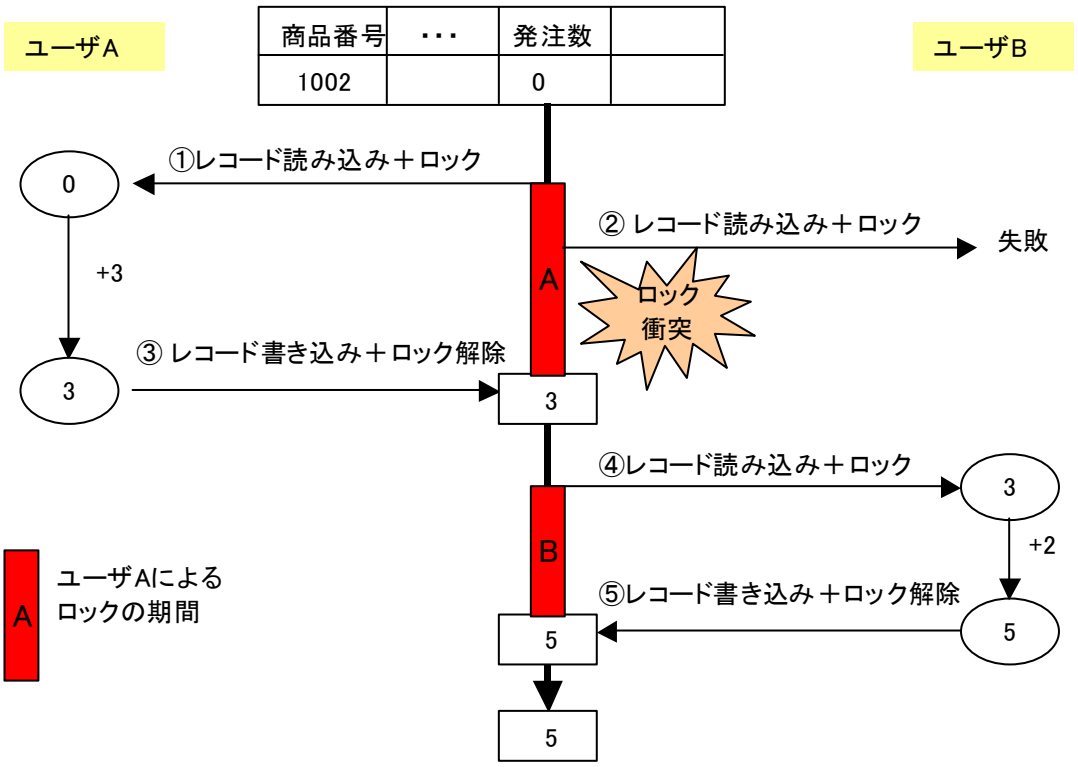


図 16-3 ロックによる更新の喪失の防止

このように、ロックを使った場合には、ユーザ A が書き込む以前にユーザ B がレコードを読み込んで、その値を元に計算を行う、ということがなくなるので、不正な更新を防止することができるようになります。

16.3. レコードロック利用時の問題

このように、ロックは平行制御の基本メカニズムですが、次に示すような問題もあります。

16.3.1. ロック待ち

まず、ロックは一人のユーザにだけ独占的なアクセス権を与えるものであり、他のユーザはロックの解除を待たせられ、その間は仕事を進められなくなる、という問題があります。
例えば、前に図 16-3 に出した例をとれば、ユーザ A がレコードを読み込みロックをかけたら、ユーザ B など他のユーザはこのレコードに対する処理を行えず、ユーザ A がロックを解除するまで待たされることになります。
ユーザ A が手際よく処理をしてくれればユーザ B の待ち時間は短くて済みますが、ユーザ A がレコードのロックをしたまま帰宅してしまったりしたら、その間ユーザ B は作業を進められなくなってしまいます。この問題の影響は、ユーザ数が多くてロックの衝突の可能性が高くなるほど深刻になってきます。

16.3.2. デッドロック

ロック待ちのもっとも深刻な形として、**デッドロック**という状態に陥ってしまうことがあります。デッドロックというのは**すくみ**とも呼ばれますが、二人以上のユーザが、お互いがお互いのロックの解除を待ったままにらみ合いになり、どちらも先に進むことができなくなってしまう状態を言います。
簡単な例として図 16-4 に、ユーザ A とユーザ B とが、商品番号#1002 と#1003 の二つのレコードを同時に更新しようとしてデッドロックになってしまうシナリオを見てみます。

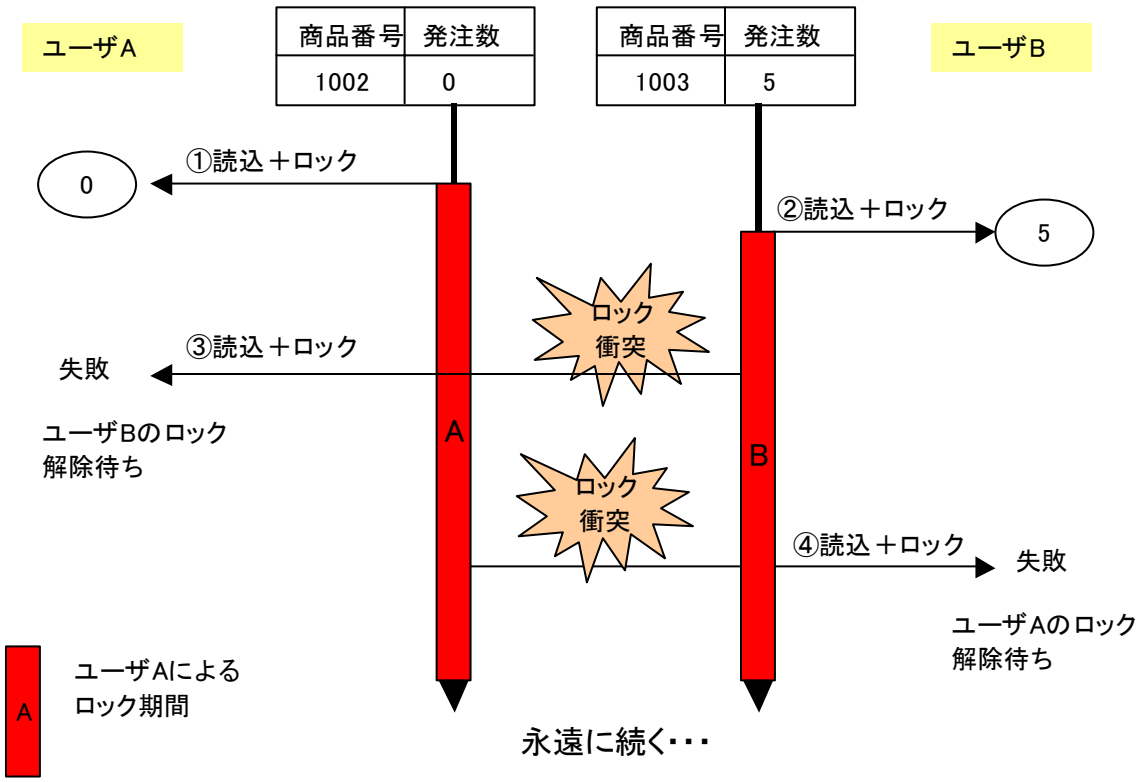


図 16-4 デッドロック

1. ユーザ A が#1002 のレコードを読み込み、ロックをかけます。

2. ユーザ B が#1003 のレコードを読み込み、ロックをかけます。ここまではいずれも成功します。
3. ユーザ A が今度は#1003 のレコードを読み込みロックをかけようとしますが、#1003 はすでにユーザ B がロックをかけているので、ロックの衝突が起こり、失敗します。このとき、ユーザ A は、ユーザ B が#1003 のロックを解除するまで待たなければなりません。
4. そこにユーザ B が#1002 を読み込み・ロックをしようとしますが、#1002 はユーザ A がロックをかけているので、ロックの衝突により失敗します。ユーザ B はユーザ A のロック解除を待たなければなりません。

この状態では、ユーザ A と B はお互いに相手が終わるのを待っているばかりで、これ以上先に進むことができなくなります。これがデッドロックです。ここでの問題は、ユーザ A が#1002→#1003 の順にロックをかけようとするのに対し、ユーザ B が逆に#1003→#1002 の順でロックをかけようとするところにあります。すなわち、デッドロックを防止するためには、設計時にレコードがロックされる順序も良く把握しておく必要があります。

16.3.3. ロックの問題を軽減するために

以上見てきたようなロックの問題は、データベースが複雑になりタイミングがクリティカルになると完全に防止することは難しいものがありますが、できるだけ問題を軽減するために、ロックを使うにあたっては、

- ロックの対象となるレコード数を極力少なくする。
- ロックをかける期間を極力短くする。

という大原則に従って、システム設計と運用を行う必要があります。

一般に、データベースのデータは相互に複雑な関連を持っているものであり、不正更新を確実に防止しようとすると多くのレコードにロックをかけなければならなくなる傾向があり、不正更新の防止とロックの最小化とは相矛盾する要求となりますので、両者を実用上問題ないように最適な設計を行うのが重要です。

以上見てきた並行制御の問題は、Magic に限らず、複数ユーザが同時に同一データをアクセスする可能性のあるすべての情報システムにおいて必ず考慮しなければならない問題です。

16.4. Magic のロック機構 1

次に、Magic では並行制御を簡単に行えるよう、さまざまな機能が用意されています。その中で基本となる、オンラインタスクでのロック機構についてここで説明します。

図 16-5 は、Magic のオンラインタスクの基本的なレコードアクセスとロックの流れを、Magic の処理レベルと対比して示したものです。

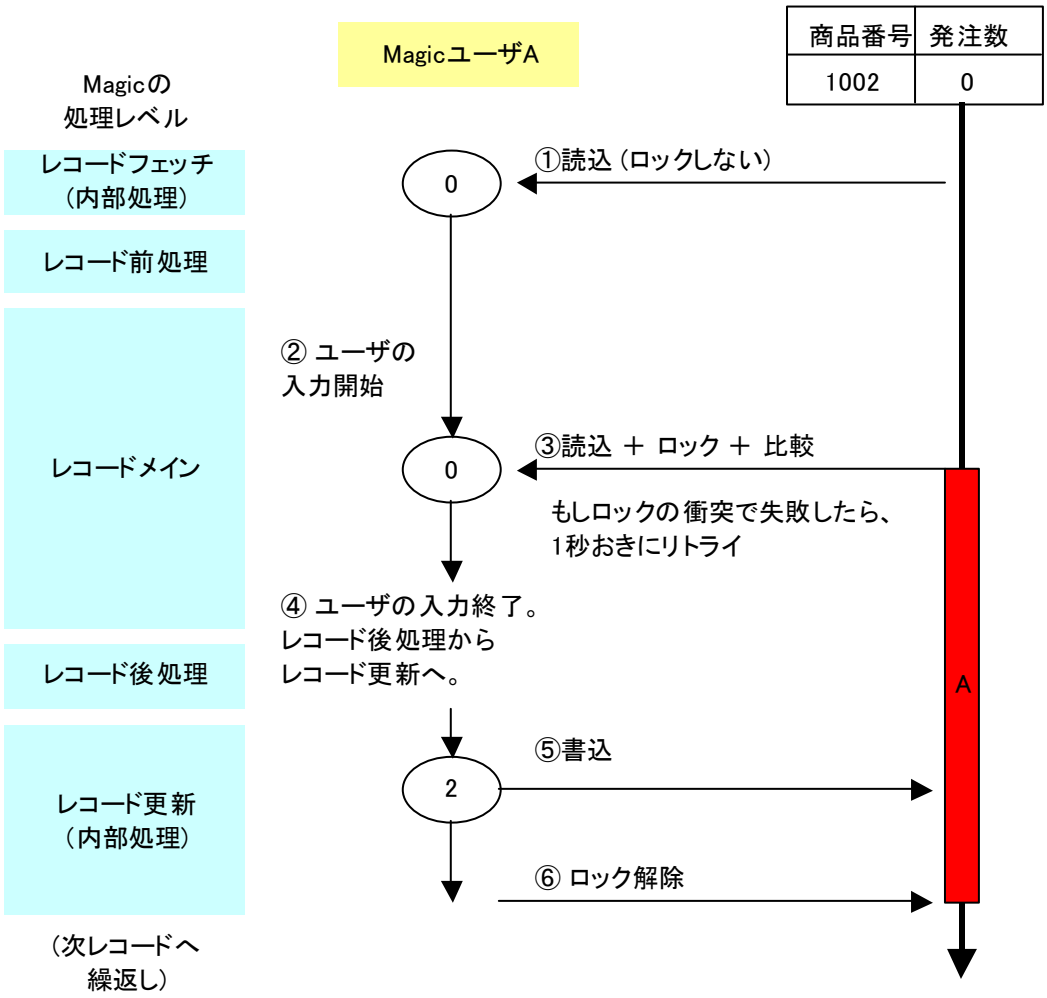


図 16-5 Magic の基本ロック機構

1. Magic は最初、DBMS からレコードを読み込みます。このときには、「ロックをかける期間を極力短くする」という原則に従い、レコードにロックをかけません。
2. レコード前処理を経て、レコードメインに進むと、Magic はユーザのキー入力を待ちます。
3. ここでユーザがキー入力を始めた瞬間、このレコードにロックをかけます。このとき同時に、レコードが他のユーザにより更新されていないかを確認するため、Magic はこのレコードを再度読み込み、最初読み込んだときと値の比較を行います。
4. ユーザの入力が終了したら、Magic はレコードメインからレコード後処理、更にレコード更新のレベルに進みます。
5. 修正されたレコードを DBMS に書き込みます。

6. 最後に、レコードのロックを解除します。

この流れが基本ですが、他のユーザが同時利用しているので、ユーザが①でロックなしで読み込んだ後、ユーザ B がロックをかけ、データを更新してしまう、ということが起こりえます。このときに起こる流れを示したのが図 16-6 です。

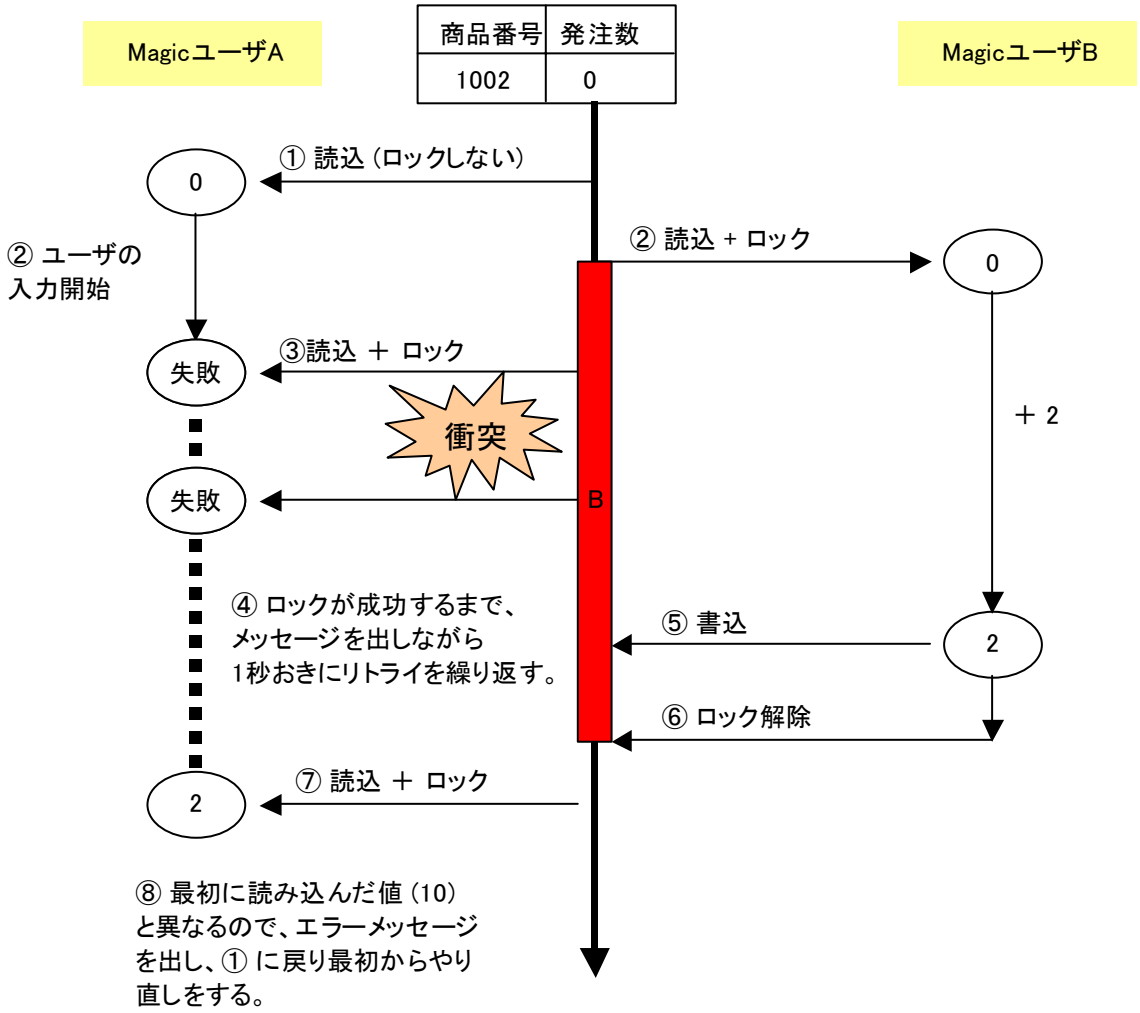


図 16-6 Magic でロック衝突が起きた場合の処理の流れ

ここでは、次のような流れとなっています。

① ユーザ A がレコードを読み込みます。最初はロックなしで読み込みます。

② 次に、ユーザ B がデータ入力を始めたので、レコードを読み込むと同時にロックをかけたとします。

③ ユーザ A の方もデータを入力したため、Magic はレコードを再度読み込むと同時にロックをかけようとしますが、B がすでにロックしているのでロックの衝突により失敗します。

④ このような場合には、Magic はステータス行に「レコードロックの解除待ちです」というメッセージを出し、1 秒おきにリトライを繰り返します。(図 16-7)

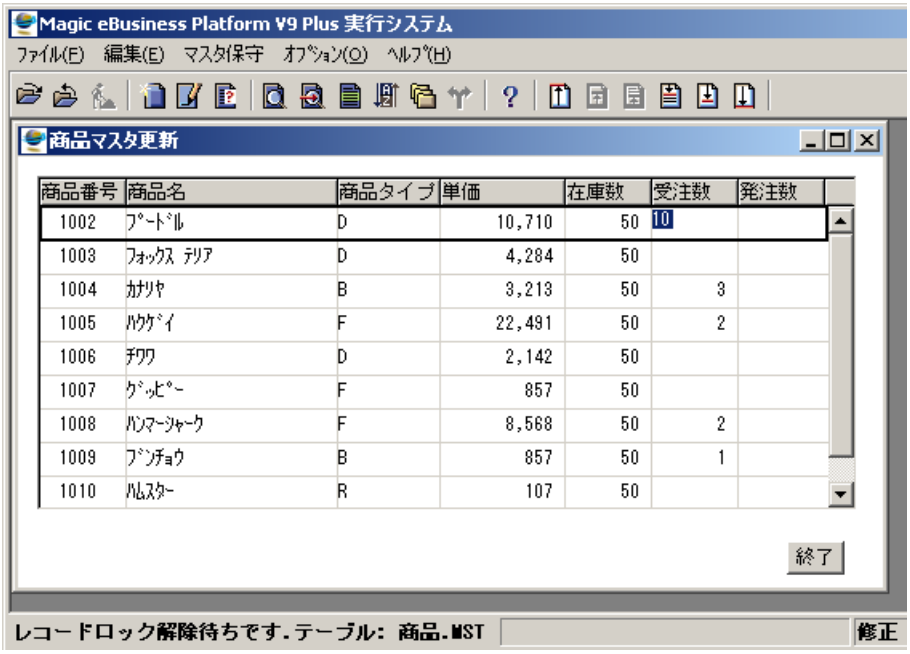


図 16-7 レコードロック解除待ちメッセージ

⑤ ユーザＢは発注数を２とし、DBMS に書き込みます。

⑥ 次いで、ロックを解除します。

⑦ ユーザＡがその後リトライすると、今度はロックが解除されているので読み込みが成功します。

⑧ ここで、Magic は値の比較をします。この例では、最新の値（2）が最初に読み込んだ値（0）と異なるので、Magic は、レコードが他のユーザ(この場合ユーザ B)によって変更されたことを検出しますので、このまま続けることはできず、ステータス行に「このレコードは他のユーザが更新しました – 再読込を行います.」というエラーメッセージを出して、新しいレコードについて ① からやり直しをします。（図 16-8）。

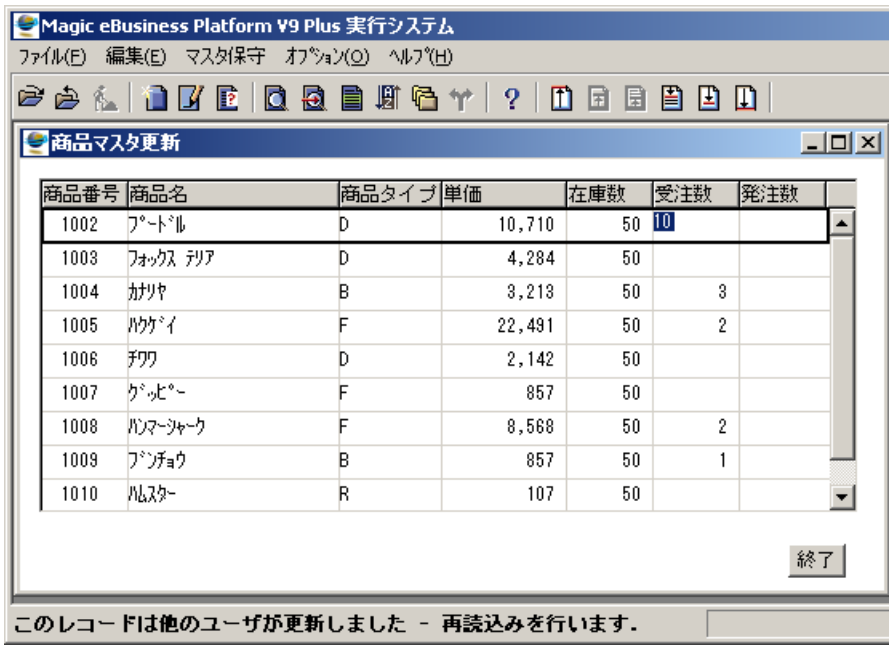


図 16-8 他ユーザによる更新の検出

このように、Magic エンジンがレコードのロックの制御を自動的にやってくれるので、開発者はこまごまとし

たロックの制御をプログラムに作りこむ必要がなくなり、プログラム開発が非常に簡単になります。

16.5. Magic で実験してみる

実際にMagicを使ってマルチユーザの機能を実験してみる前に、いくつか補足が必要なことがらがあるので、説明します。

16.5.1. Magic プログラムのトランザクション設定

マルチユーザ環境で使う場合、ロックのほかにトランザクションの設定で動作が大きく変わります。トランザクションの設定は Magic のタスク特性でタスクごとに設定することができ、デフォルトでは**遅延トランザクション**という、Magic 独自の便利なトランザクション機能を利用するようになっていますが、これはトランザクションの機能をよく理解した後でないと活用が難しいので、本章で使うプログラムでは遅延トランザクションを含めトランザクションを一切使わずにプログラムを作ります。

動作環境： トランザクションを一切使わないようにするには、まず、環境設定を次にようにして確認してください。

1. Magic のアプリケーションを閉じ、メニュー 設定(S) → 動作環境(E) で動作環境ダイアログを開きます。
2. マルチユーザ(M) タブを開きます。
3. 次の設定を確認してください（図 16-9)。デフォルトでこのような設定になっているはずです。
   - マルチユーザアクセス = Yes
   - ISAM トランザクション = No
   - ロック前にトランザクション開始(ISAM) = No

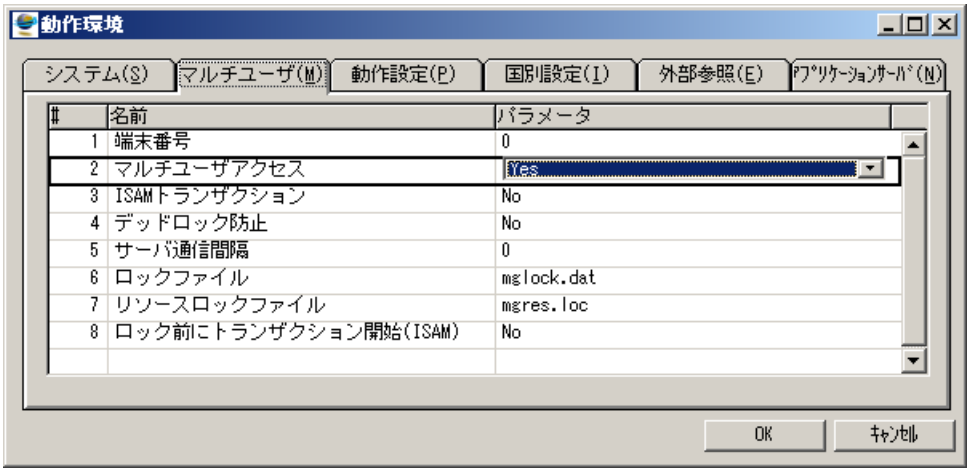


図 16-9 動作環境の設定

タスク特性： 次に、各タスクごとに、タスク特性のトランザクションモードを変更します。

1. アプリケーションを開いてプログラムリポジトリを開きます。
2. 実行しようとするプログラム(例えば「商品マスタ更新」)をズームして開きます。
3. Ctrl+P でタスク特性を開きます。
4. 拡張(E)タブを開きます。
5. トランザクションモードが、最初は「D=遅延」になっているので、「P=物理」に変更します（図 16-10)。

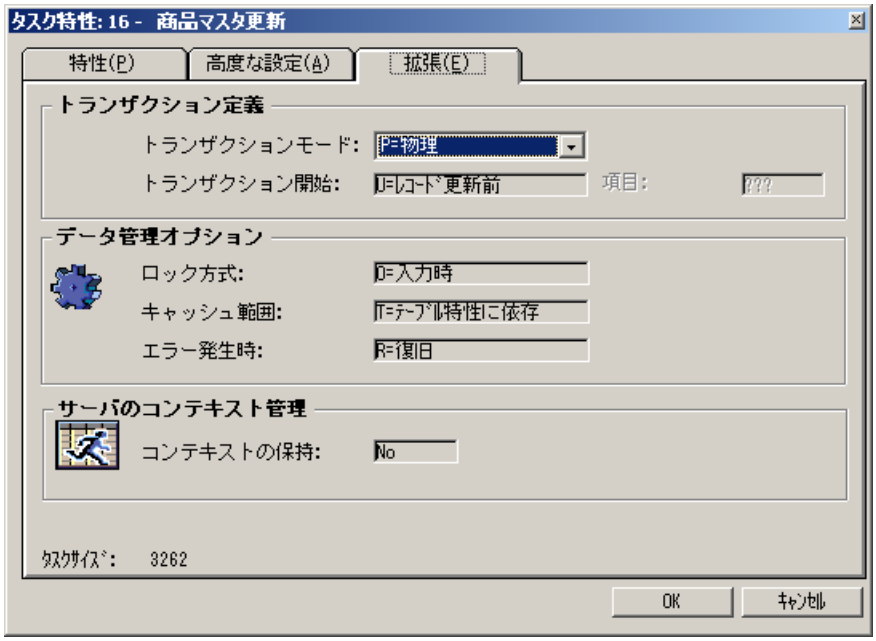


図 16-10 タスク特性のトランザクションモードを物理に設定

6. タスクを閉じます。

このような設定変更を、アプリケーション中のすべてのオンラインタスクに対して行ってください。

参考：

- トランザクションモードが「D=遅延」のままだと、レコードロックがかかりません。
- バッチタスクでは、デフォルトでトランザクションモードが「物理」になっているので、変更の必要はありません。

16.5.2. 同一アプリケーションを開く

次に、マルチユーザの実験環境を実際にどのように作るかです。
エンドユーザの運用環境では、通常は各ユーザが 1 台づつ PC を使っており、各 PC に Magic がインストールされ、データベースサーバにある DBMS をネットワークで共有してアクセスする、という形態が一般的です。
しかし、開発時やテスト時には、そのような環境を用意するのは、設備上面倒だったり難しかったりします。できれば、一台の PC 上ですべてのテストを行いたいものです。Magic では、1 台の PC 上で複数起動することができるので、少人数での開発・テスト段階では、1 台の PC で複数起動して、DBMS もローカルにインストールされているものを使うことにより、あたかもマルチユーザ環境で実行しているかのように開発・テストすることが可能です。
ここでひとつ注意すべきことは、複数起動した Magic で同一のアプリケーションファイル（MCF ファイル）を開く場合、**開発版**ではなく**実行版**で実行する必要がある、ということです。
Magic をインストールしたディレクトリ（デフォルトでは、C:¥Program Files¥Magic¥eDeveloperPlusTrial）を開いてみてください。ここには Magic の実行に必要となるモジュールが格納されていますが、その中に mggenw.exe と mgrntw.exe という実行可能モジュールがありますが(図 16-11)、mggenw.exe が開発版モジュールで、mgrntw.exe が実行版モジュールです。体験版には、この両方のモジュールが入っています。

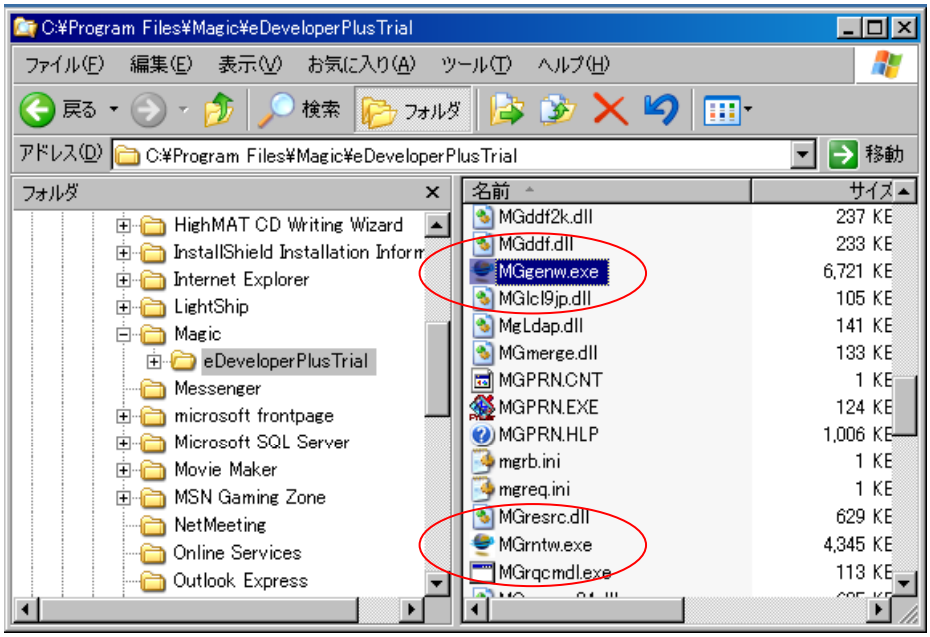


図 16-11 開発版モジュール mggenw.exe と実行版モジュール mgrntw.exe

開発版は、今まで使ってきたもので、Magic アプリケーションを開発するときに使い、Magic アプリケーションの開発・修正と、実行の機能の両方をサポートしています。実行版は、Magic アプリケーションを実行する機能だけを持っており、アプリケーションを修正することはできません。

開発版を複数起動した場合、同一のアプリケーション（MCF ファイル）を同時に開くことができません[4]。開こうとすると、後から開こうとした方で、「テーブルのロック待ちです.: ･･･.MCF」というエラーがステータス行に表示されます。Magic は繰り返しリトライを続けるので、キャンセルする場合には ESC キーを押します。

図 16-12 開発版で同一 MCF を開いた場合のエラーメッセージ

これは、開発版ではアプリケーションを修正するので、整合性を保つため、ファイル全体に排他的なロックをかけるからです。

一方、実行版では、アプリケーションの修正を行わないので、ファイルに共有ロックをかけます。共有ロックなので、実行版同士ならば共有してオープンすることができます。

以上をまとめたのが、図 16-13 です。

このため、マルチユーザ環境のテストのために、同一アプリケーションを複数の Magic でオープンする場合には、実行版 mgrntw.exe を起動するようにしてください。(mgrntw.exe へのメニューはないので、エクスプローラやコマンドラインから起動するか、ショートカットを作成してください)。

[4] Magic の備える**チーム開発**の機能を用いれば、同一 MCF ファイルを複数の開発版で開くことができますが、設定がいろいろと煩雑になるので、ここでは簡単にできる実行版での実行で話を進めていきます。

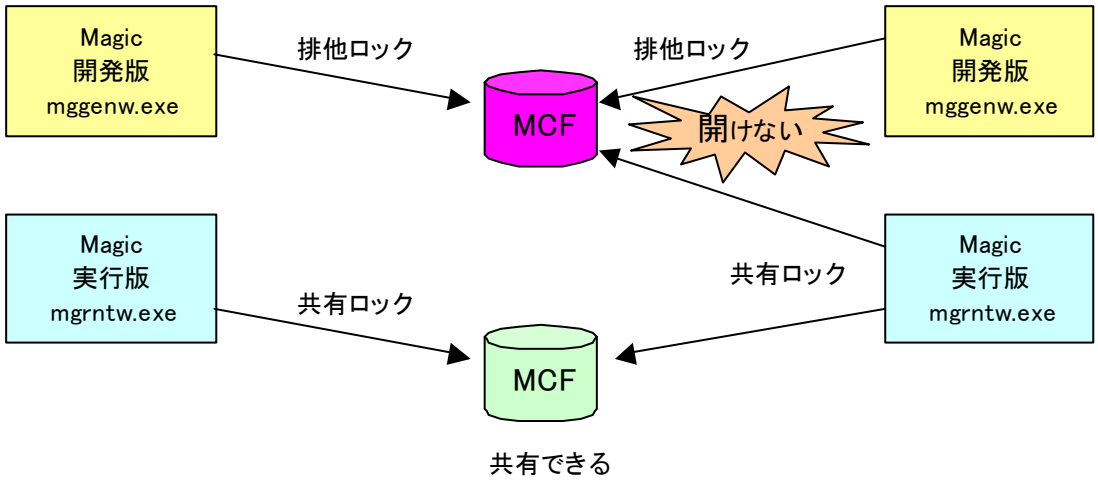


図 16-13 MCF の共有

16.5.3. 実行してみる

それでは、実際に実行してみて、Magic のロック機構を試してみましょう。

最初に、Magic の実行版を二つ起動します。

1. Explorer などで、Magic のディレクトリから mgrntw.exe を起動します。
2. petshop アプリケーションを開きます。
3. メニューから、「商品マスタ更新」プログラムを開始します。
4. 1～3 をもう一度繰り返します。

これにより、Magic 実行版が二つ起動し、それぞれで商品マスタ更新プログラムが実行されます（図 16-14）。

以下の説明では、二人のユーザを想定して、左側の Magic を**ユーザ A**、右側のを**ユーザ B** と呼びます。

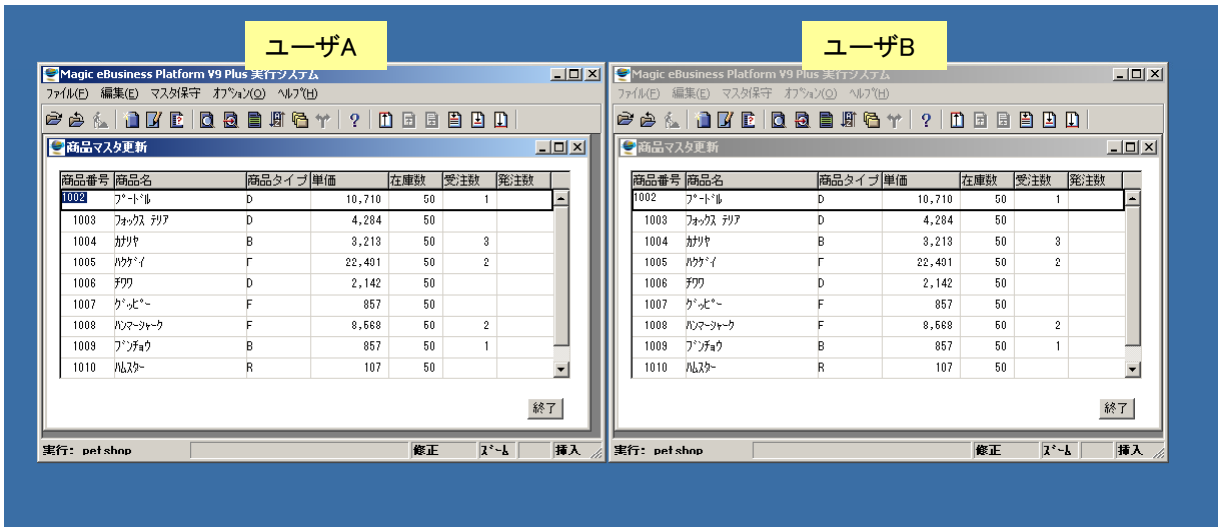


図 16-14 実行版を二つ起動し、商品マスタ更新プログラムを開始する。

次に、レコードロックがちゃんとかかるかを見てみましょう。図 16-6 Magic でロック衝突が起きた場合の処理の流れの図と対照しながら、流れを追ってください。

1. 最初は、ユーザ A、ユーザ B ともに、商品番号#1002 のレコードにカーソルがあり、発注数は 0 です。
2. ユーザ B で、「発注数」カラムで「2」と入力します。このとき、目には見えませんが、このレコードにはロックがかけられます。カーソルが別のレコードに移動するとロックは解除されてしまうので、カーソルは「発注数」カラムから動かさずこのままで次のステップに進んでください。
3. 次に、ユーザ A で、同じ#1002 のレコードの「発注数」に「3」を入力します。→ 入力しようとした瞬間に、Magic は再読込をしてロックをかけようとしますが、すでにユーザ B がロックをかけているのでロックの衝突となります。このため、「レコードロック解除待ちです」というメッセージが、ステータス行に表示されます。

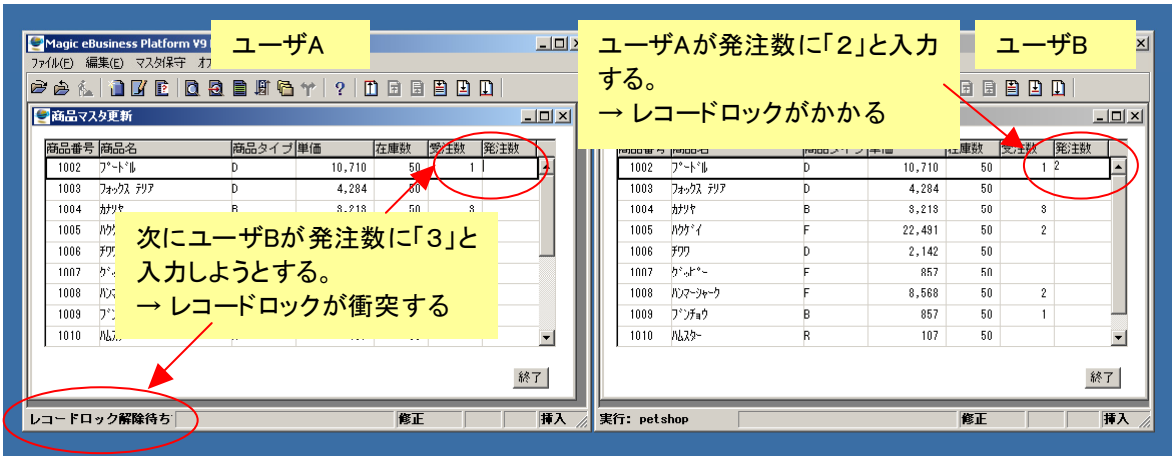


図 16-15 レコードロックの衝突

4. ユーザ B が、カーソルを次の行に移動します。→ レコードが書き込まれ、ロックが解除されるので、ユーザ A の側でロック待ちが終わります。しかし、ユーザ A が再読込したレコードでは発注数が 2 に変わっていて、最初の値 0 と異なるので、「このレコードは他のユーザが更新しました」のエラーメッセージがステータス行に出て、最初からやりなおしとなります。このときには、発注数は最新の値 2 になっています。

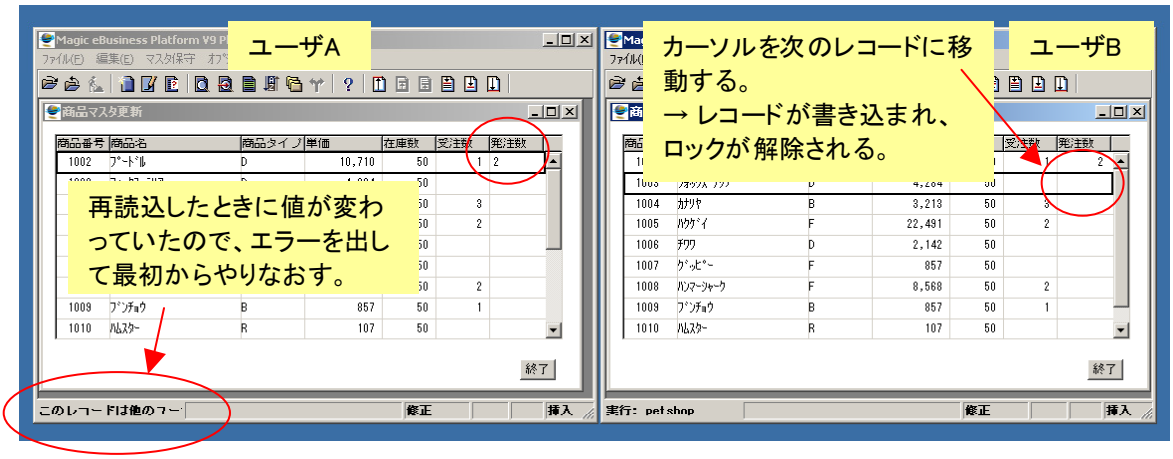


図 16-16 ユーザ B が更新したため、ユーザ A はやりなおしとなる。

以上が、Magic の基本的なロック機構です。

16.6. Magic のロック機構 2

これまでは、Magic のロック機構の基本として、オンラインタスクで典型的な(デフォルトの)ロックの設定で説明しましたが、Magic のロック機構はその他にもさまざまな設定があります。ここではロックのバリエーションについて説明するとともに、いくつかの補足事項も説明します。

16.6.1. ロックのタイミング

先の例のオンラインプログラムでは、ユーザがキー入力した瞬間にレコードロックがかかるようになっていましたが、それ以外のタイミングでロックをかけたり、あるいは全くロックをかけないようにすることもできます。

ロックのタイミングの制御は、タスク特性の「拡張(E)」タブにある「ロック方式」パラメータにより設定されます（図 16-17)。

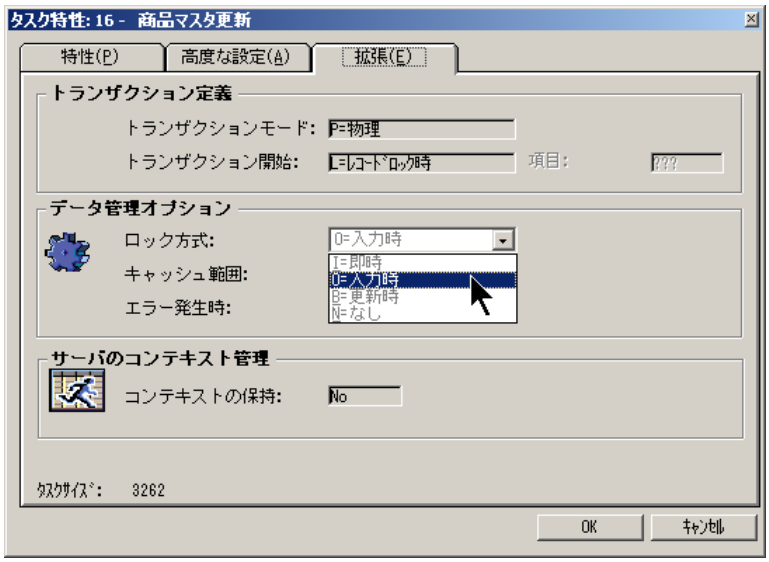


図 16-17 ロック方式パラメータ

このパラメータでは、次のような設定を行うことができます。

ロック方式	意味
I=即時	カーソルがレコードに移動するとすぐ、ユーザの入力に関わりなくロックをかける。
O=入力時	(デフォルト) ユーザがキー入力を始めた瞬間にロックをかける。
B=更新時	レコード後処理の後、DBMS にレコードを書き込むときに至ってからロックをかける。
N=なし	ロックは一切かけない。

表 16-1 ロック方式パラメータの意味

オンラインプログラムでは、通常デフォルトの「O=入力時」が最適ですが、特殊な要件のためにタイミングを変える場合には、このパラメータを変更します。

16.6.2. バッチタスクの場合

バッチタスクの場合には、ユーザのキー入力がなく、各レコードごとの処理時間も通常は極めて短いので、デフォルトのロック方式は「I=即時」となります。。オンラインプログラムを開発版で開き、「タスクタイプ」を

「O=オンライン」から「B=バッチ」に変更すると、Magic は自動的にロック方式を I=即時に変更します。

16.6.3. リンクレコードへのロック

メインテーブルのレコードにロックがかかるタイミングで、リンクコマンドによりリンクされているレコードに対してもロックがかかります。

例えば、受注入力の親タスクでは、メインテーブルが受注テーブルですが、これに制御テーブル、顧客テーブルがリンクされています。図 16-18 では、受注番号#101 のレコードに対し、顧客番号#1008 の顧客レコードがリンクされています。また、画面では見えませんが、キーの値が 1 の制御テーブルのレコードもリンクされています。受注#101 のレコードがロックされるのと同時に、これらのレコードもロックされます。

図 16-18 受注入力画面

また、同様に、受注入力の子タスクでは、メインテーブルが受注明細テーブルで、それに商品レコードがリンクされています。上の例では、受注番号#101 の明細#1 には商品#1002 がリンクされているので、子タスクでロックがかけられる場合には、受注明細のレコード#101 とともに、商品#1002 のレコードもロックされます。

16.6.4. リンクレコードへのロックの制御

場合によっては、リンクされたレコードにロックをかけないようにしたい場合があります。例えば、読み込み専用のレコードであればロックする必要はありません。不必要なロックを減らすのは、マルチユーザ環境で円滑な運営をするための大原則なので、ロックの対象とするレコードはきめ細かく制御する必要があります。

特定のテーブルへのリンクレコードのロックを行うか否かは **DB テーブル**の「アクセス」パラメータにより指定できます。DB テーブルは、以下のようにして開きます。

1. プログラムリポジトリからタスクを開きます。
2. メニュー「タスク環境(K)」→「DB テーブル(D)」（あるいは、Ctrl＋D）を選びます。→ DB テーブルが開きます。（図 16-19）。DB テーブルには、このタスクで参照しているテーブル(メインテーブルおよびリンクテーブル)の一覧が表示されます。この一覧は Magic が自動的に管理しています。

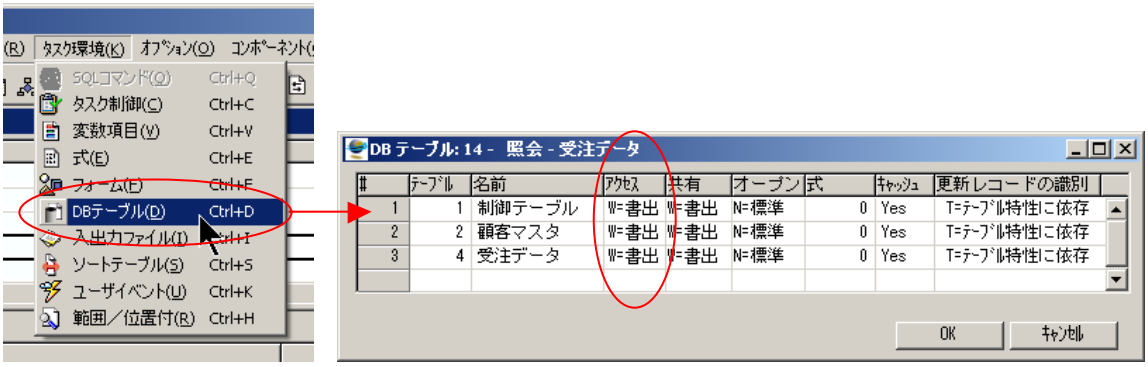


図 16-19 DB テーブルの「アクセス」パラメータ

「アクセス」パラメータは、すべてのテーブルに対してデフォルトで「W=書出」となっています。これは、レコードの更新を行う可能性がある、という意味です。

特定のテーブルに対してレコードロックを行わないようにしたい場合には、アクセスパラメータを「R=読込」に設定します。これは、このタスクでは、このテーブルのレコードの更新を行わない、つまり読み込み専用でこのテーブルを使う、という意味で、R=読込になっているテーブルに対しては、Magic はレコードロックを行いません。

マルチユーザ環境で起こるレコードロックの衝突を回避するために、R=読込の設定にすることが良く行われます。例えば、制御テーブルのロックの衝突を避けるために、制御テーブルのアクセスパラメータを「R=読込」にします。

もちろん、アクセスパラメータを変更した場合には、いろいろとプログラムを修正する必要があります。例えば、制御テーブルはレコード後処理で項目更新コマンドで更新しているので、R=読込にしただけだと、実行時にエラーとなります。読込専用なのにデータを更新しようとしたからです。

エラーを避けるためには、このタスクではレコードを更新しないようにしなければなりません。通常は項目更新コマンドで更新する代わりに、制御テーブルのレコードの更新を行うバッチタスクを別途作成しておき、このバッチプログラムを呼び出すことにより更新を行うようにする、という手法をとります。DB テーブルの設定の有効範囲はタスク単位なので、別タスクを立ててやれば更新することが可能となります。

16.6.5. タスクのモードとの関係

レコードロックは、レコードを変更する可能性のある場合にだけ必要となります。従って、タスクが修正モードの場合には、今まで述べてきた条件に従ってレコードロックがかかります。その他のモードの時には、次のようになります。

- **照会：** タスクのモードが「照会」の場合には、すべてのレコードは読込専用で、データの修正が起こらないため、DB テーブルの設定にかかわらずレコードロックはかかりません。
- **登録：** タスクが登録モードの場合には、メインテーブルのレコードにはロックがかかりません。登録モードの場合には、まだレコードが DBMS に作成されていないので、ロックすべきレコードがないからです。ただし、登録モードであっても、リンクされたレコードにはロックがかかります。
- **削除：** バッチタスクでは、初期モードとして「D=削除」というものがあります。これは、メインテーブルのレコードのうち、範囲指定などの条件に合致するレコードを 1 件づつ削除するモードです。削除モードでは、修正モードと同様にレコードロックがかかります。

16.6.6. テーブルレベルのロック

Magic では、レコードレベルのロックのほかに、テーブルレベルでロックをかけることもできます。テーブルレベルでロックをかけると、他のユーザがそのテーブルをオープンすることさえ禁止することができるようになります。集計や大量の更新を行う際に、他のユーザが処理中にレコードを変更して一貫性を崩してしまうようなことがないようにしたい場合などに使います。

テーブルロックの設定

テーブルレベルのロックは、DB テーブルの「共有」パラメータによって行います。

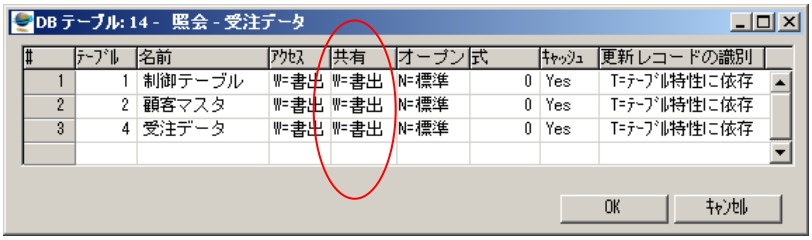


図 16-20 DB テーブルの「共有」パラメータ

共有パラメータの値は、W=書出し、R=読込、N=なしの 3 つです。アクセスパラメータと「N=なし」以外は同じですが、アクセスパラメータが「自分がどういう形でアクセスするか」を指定するのに対し、共有パラメータは「他のユーザがどういう形でアクセスするのを許可するか」を指定するもので、意味は以下の通りです。

共有パラメータ	意味
W=書出	他のユーザがデータの修正をしても良い。
R=読込	他のユーザは、データの読込はしてもよいが、書込みは許可しない。すなわち、読込専用でだけ許可する。
N=なし	他のユーザに一切のアクセスを許可しない。排他的にテーブルを独占したい場合に使います。

表 16-2 「共有」パラメータの意味

テーブル共用の可否

ユーザ A があるファイルをオープンした後、ユーザ B が同じファイルをオープンしようとしたとき、ユーザ B がオープンできるか否かは、ユーザ A の共有パラメータとユーザ B のアクセスパラメータの組み合わせで、以下のように決まります。

ユーザ A の共有　→	W=読込	R=書込	N=なし
ユーザ B のアクセス　↓			
W=書込	○	×	×
R=読込	○	○	×

表 16-3 共有とアクセスの組み合わせによるファイルオープンの可否

テーブルロックとレコードロックの関係

次に、DB テーブルでのテーブルロックの設定と、レコードロックとの関係について考えて見ます。
レコードロックというのは、レコード単位で他のユーザとの共用を制御するものであり、共用するユーザの両方がレコードを変更する可能性がある場合にのみ必要となります。逆に言うと、レコードの変更の可能性のない場合、あるいは変更しても最大でも一人だけ、という場合には、レコードロックは不要となります。レコードロックは DBMS に対するアクセスを行い、オーバーヘッドを伴う処理であるので、不要なレコードロックは極力避けるようにしなければなりません。
DB テーブルのアクセス、共有パラメータの意味を考えると、レコードロックは次のような場合には不要となります。

- アクセスパラメータが「R=読込」の場合には、読込専用でレコードを修正することがないのだから、レコードロックをかける必要はない。
- 共有パラメータが「N=なし」の場合には、テーブルを自分だけが排他的に利用するので、他のユーザは一切アクセスできない。従って、この場合もレコードロックは不要である。
- 共有パラメータが「R=なし」の場合には、他ユーザに対してテーブルの読込は許可するが書込みは許可しない。従って、他のユーザがレコードを変更する可能性はないので、レコードロックは不要である。

このようなケースを消去してゆくと、結局、レコードロックが必要となるのは、アクセスが「W=書込」かつ共有が「W=書込」の場合のみ、ということになります。

17. 受注入力プログラムの場合

前章では、Magic のロック機構の基本について説明しましたが、例としてはテーブルを一つ使っただけのごく簡単なオンラインプログラムだけでした。このような簡単なマスターテーブル更新プログラムなどでは、プログラムに手を加えなくとも、Magic のロック機能のみに頼って十分な排他制御ができました。

しかし、複数のテーブルをリンクしたり、受注入力のように親子のタスクで 1 : N 関係のレコードを扱う場合などでは、**並行性**（複数のユーザが、ロック待ちなどにより中断させられてしまうことが極力なく、同時並行してデータをアクセスできる度合い）を確保するために、プログラムを変更する必要が出てきます。

本章では、マルチユーザ環境を意識せずに作成した受注入力プログラムを二人のユーザが同時に実行した場合に起こるロックの問題を具体的に見ていき、プログラムをどのように修正すればよいかを説明します。

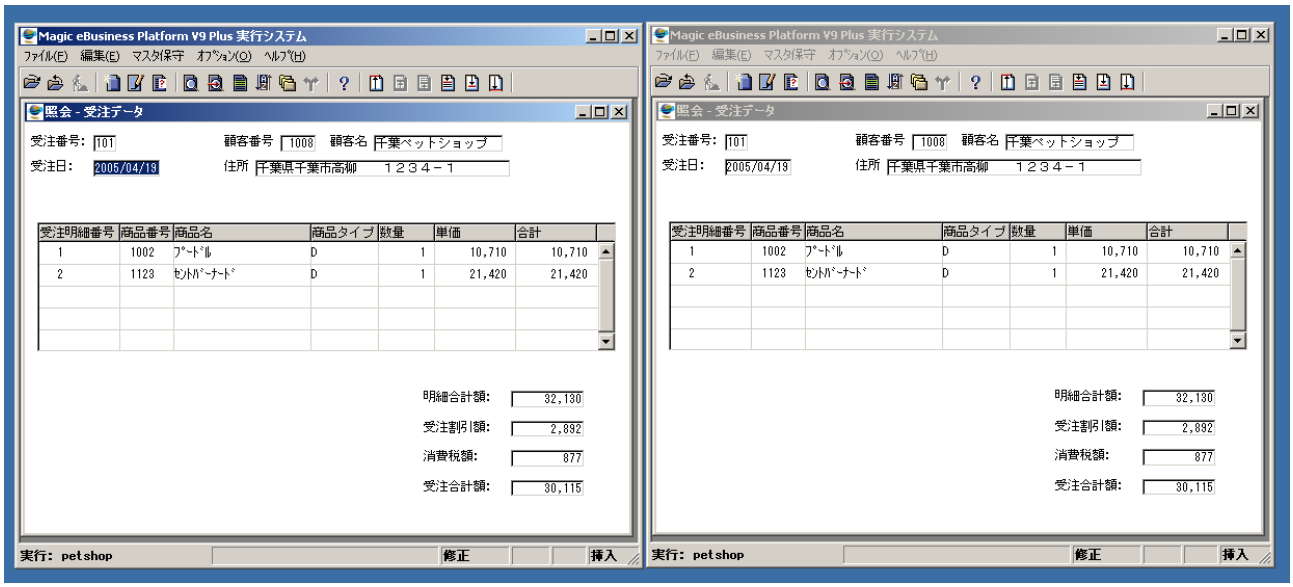


図 17-1 受注入力プログラム

17.1. 準備

受注入力プログラムの実験を始める前に、プログラムのトランザクションの設定を変更しておきます。前に書いたように、Magic のオンラインプログラムでは、デフォルトで遅延トランザクションという Magic 独自のトランザクション機能を使うようになっていますが、ここではロックの機能を理解するために、トランザクションは一切使わない設定にします。

1. petshop アプリケーションを開いて、受注入力のプログラム（プログラム 14 番「照会 – 受注データ」）を、プログラムリポジトリの最後にコピーします。名前は「受注入力２」としてください。直接もとのプログラムを修正しても良いのですが、やり直しができるように、念のためににコピーを作って修正します。

   参考： プログラムのコピーには、カーソルをプログラムリポジトリの最後に置いてから、「編集(E)」メニュー → 「複写登録(Y)」を使います。

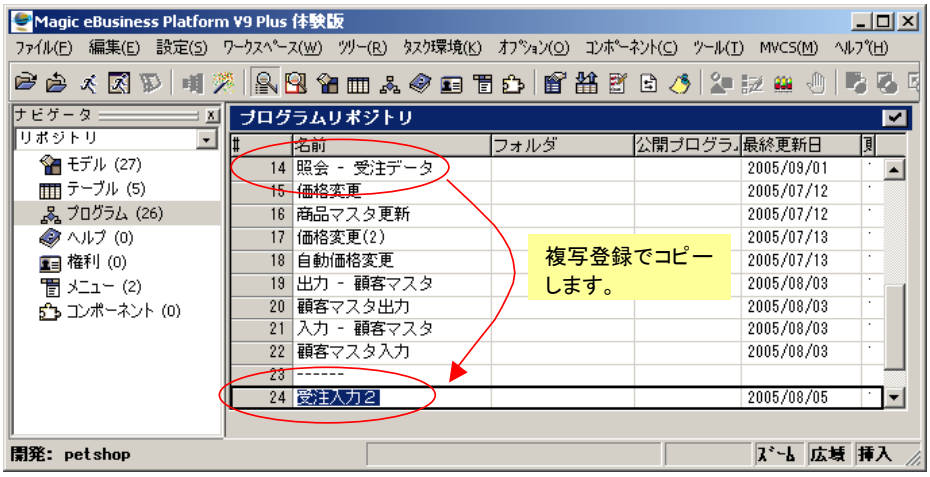


図 17-2 受注入力プログラムをコピーする。

2. コピーした受注入力２プログラムを開いてください。
3. タスク特性を開き（Ctrl+P で開きます）、「拡張(E)」タブを開きます。

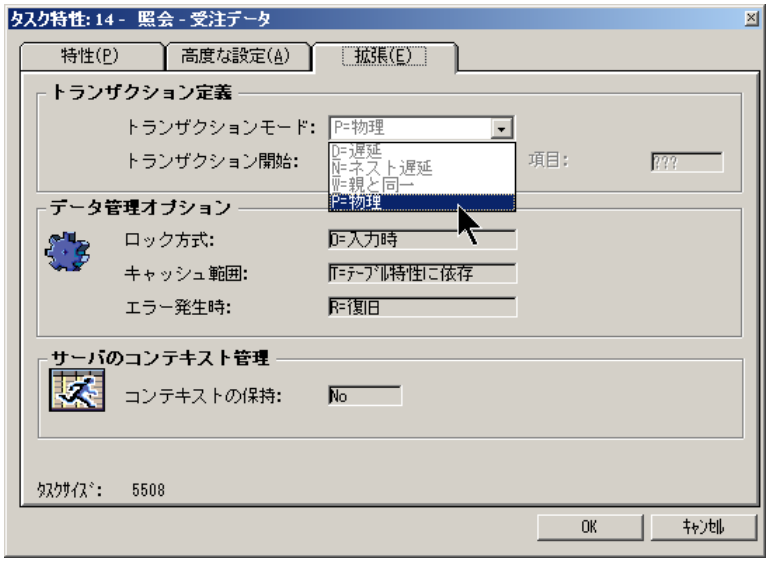


図 17-3 タスク特性のトランザクションモード設定

4. 「トランザクションモード」が「D=遅延」になっていたら、「P=物理」に変更します。

参考： この設定については、前章の「Magic プログラムのトランザクション設定」の節を参照してください。

5. あと、細かなことですが、「受注入力」という名前にふさわしいように、プログラム起動時からデータ入力をできるよう、初期モードは「C=登録」にしましょう。

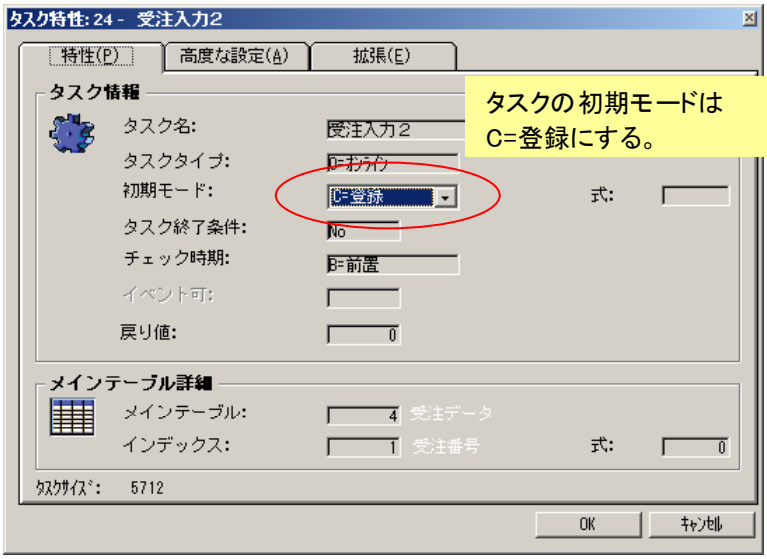


図 17-4 タスクの初期モードを登録にする

6. 最後に、実行版でこのプログラムを呼び出せるように、メニューに登録します。メニューリポジトリを開いて、「基本コンテキストメニュー」の最後にこのプログラムを追加します。

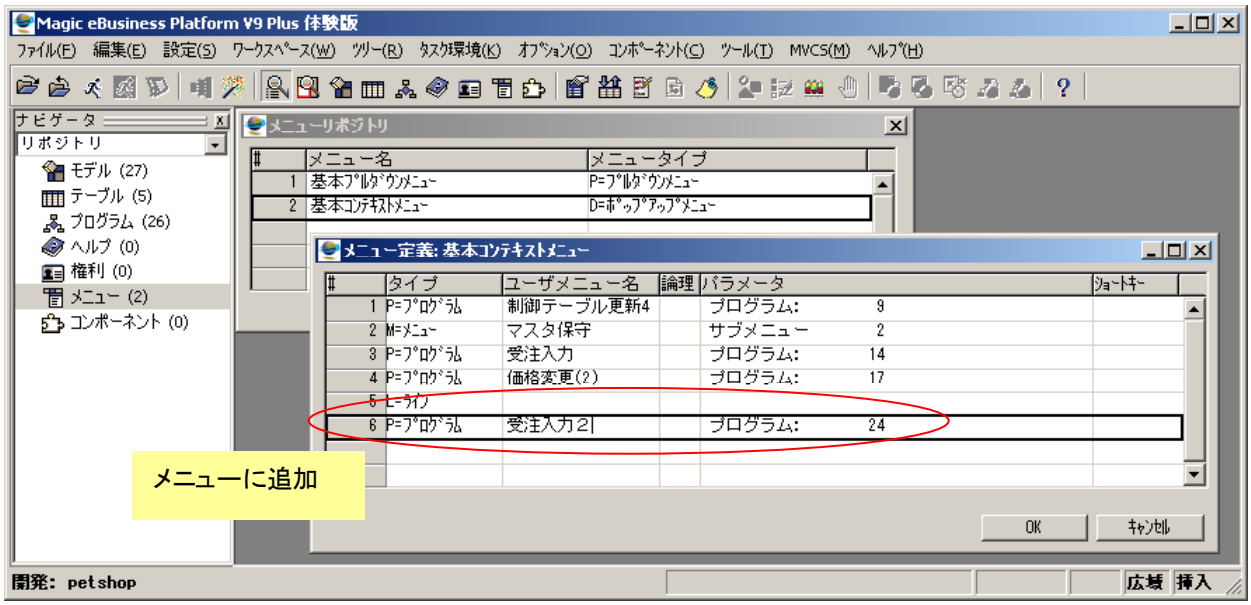


図 17-5 メニューにプログラムを追加

以上で準備は終了です。

なお、以下の説明ではテスト→プログラム修正→再テスト→・・・ というサイクルで話が進んでいきます。プログラムの修正は開発版（体験版、mggenw.exe)で行いますが、テストでは二つの Magic プロセスが同一アプリケーションを同時にアクセスすることになるので、実行版（mgrntw.exe）を二つ立ち上げて使います。前章で説明したように、開発版と実行版とを使って同時に同じアプリケーションを開くことはできないので、

- 開発版でプログラムを修正しているときには、実行版はアプリケーションを閉じておく。
- 実行版でテストをしているときには、開発版ではアプリケーションを閉じておく。

というやりかたで行うようにしてください（図 17-6）。

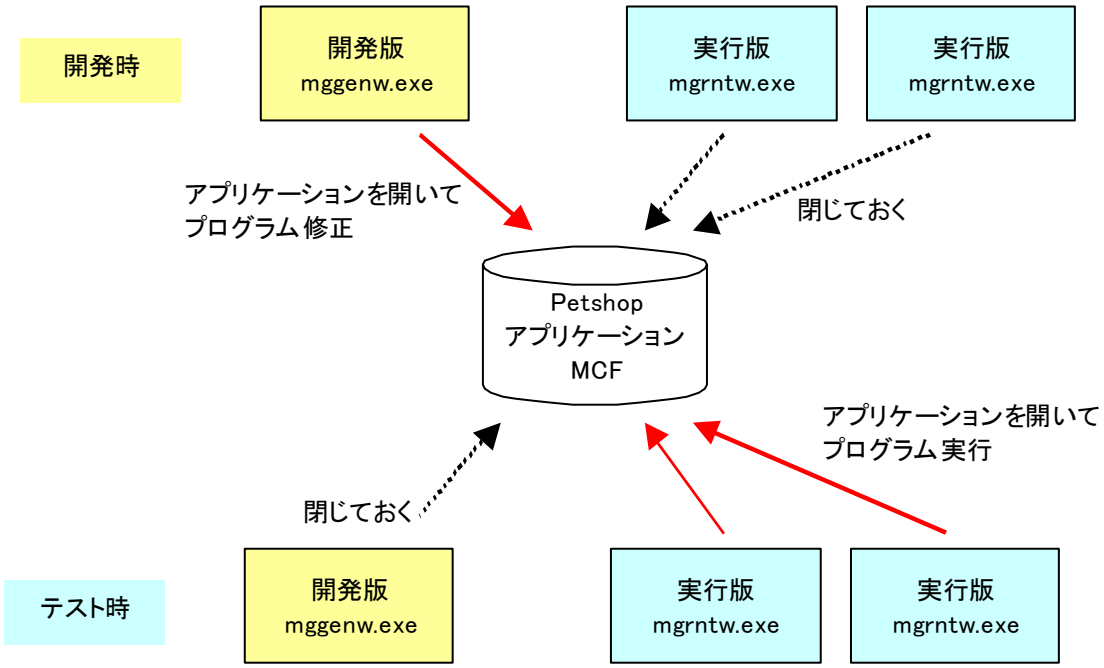


図 17-6 開発時とテスト時のアプリケーションオープンのやりかた

17.2. 制御テーブルのロック衝突

17.2.1. 制御テーブルのロック衝突の問題

それでは実験を開始しましょう。

1. 前述のように、開発版ではアプリケーションを閉じ、実行版では二つとも petshop アプリケーションを開きます。
2. 右クリックメニューから、先ほど登録した「受注入力２」を起動します。(両方とも)。

初期画面では、図 17-7 のようになります。初期モードが登録モードなので、既存のデータは表示されず、空のフォームが表示されます。前章の例にならって、左側をユーザ A、右側をユーザ B と呼びます。

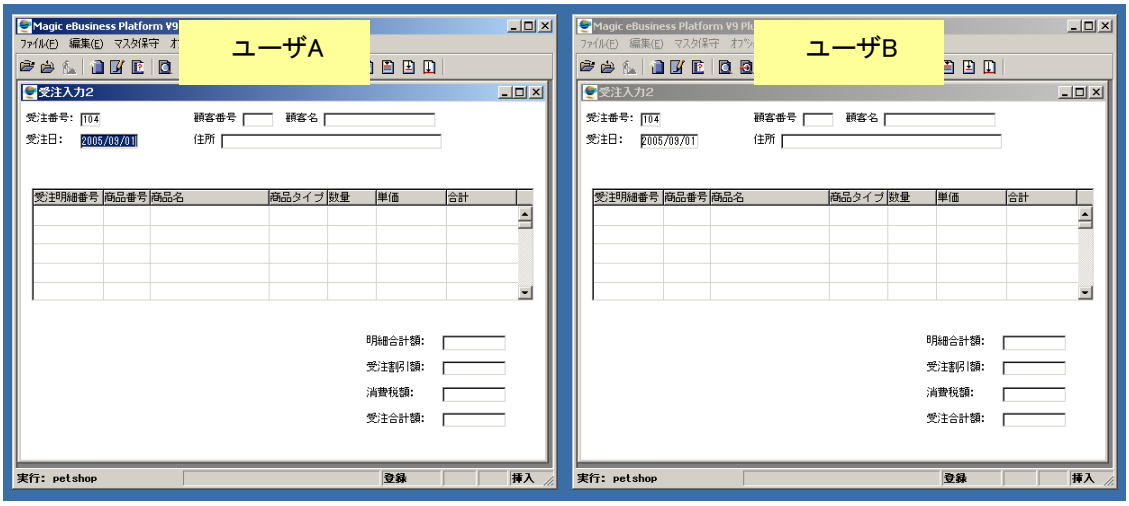


図 17-7 受注入力の初期画面

ここで画面を良く見てみると、「受注番号」が両ユーザとも、104 になっています。受注テーブルのデータの状態によっては、104 以外になっているかもしれませんが、いずれにしても同じ数値になっているはずです。ユーザ A とユーザ B は別の受注データを入力しようとしているのですから、このことは、異なる受注データに対し、同じ受注番号を発番してしまっている、ということを意味します。
このへんからまず問題が見られるのですが、とりあえず次に進んでみましょう。

3. ユーザ A で、顧客番号として#1008（千葉ペットショップ）を選択します。直接数値を入力しても良いし、ズームして顧客一覧から選択しても構いません。Tab キーを押すと、子タスクの方にカーソルが移動します。
4. 次に、ユーザ B で、顧客番号として#1234（ペットセンター神田）を選択します。すると、ユーザ B の方で、「レコードロックの解除待ちです： 制御.DAT」というエラーがステータス行に表示されます（図 17-8)。

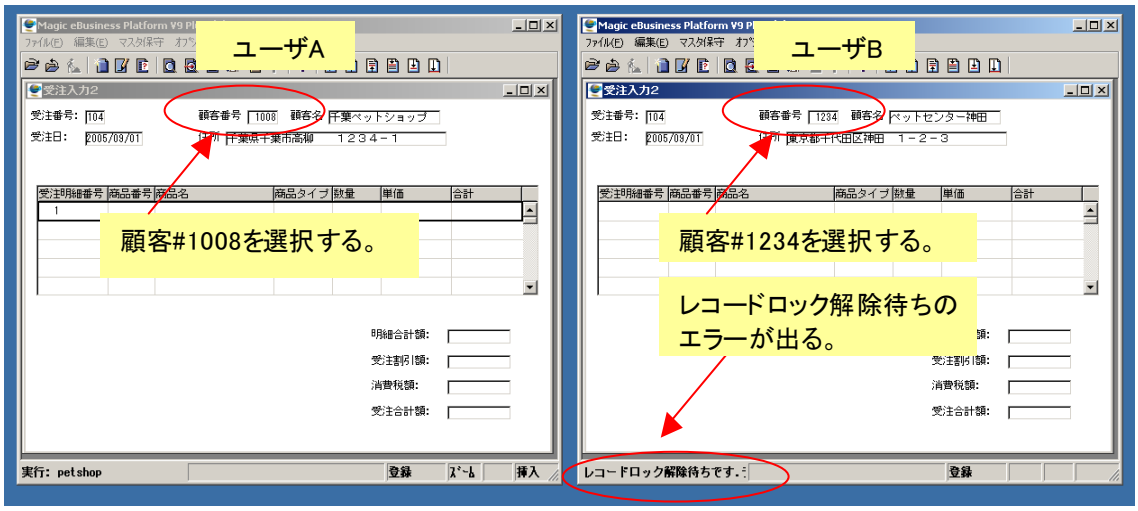


図 17-8 制御テーブルのレコードロック解除待ち

これはいつまで続くのでしょうか？もう少し進んでみます。

5. ユーザ A で、商品#1002（プードル）を選び、個数は 1 とします。
6. ESC キーを押すと、カーソルが親タスクに戻ります。この時点ではまだユーザ B のレコードロック解除待ちは続いています。
7. もう一度 ESC キーを押すと、親タスクが終了します。ここで始めてユーザ B のレコードロックが解除されます。

これでは、ユーザ A が受注入力を行っている間、ユーザ B はレコードロック解除待ちとなってしまい、仕事になりません。
なぜこのようなことが起こるのでしょうか？
レコードロックの衝突が起こっているのは、ステータス行のエラーメッセージにあるように、「制御.DAT」テーブルです。これは、レコードメインでリンクコマンドによりリンクされているテーブルです。
ユーザ A で顧客番号が入力されると、レコードロックがかかりますが、このときにロックのかかるレコードは、以下のようになります。

	テーブル名	レコードロック
メインテーブル	受注データ	登録モードなので、レコードロックはかからない。
リンクテーブル	制御データ	キー値が 1 のレコードがリンクされており、このレコードがロックされます。
	顧客データ	顧客番号 1008 のレコードがリンクされており、このレコードがロックされます。

表 17-1 ユーザ A のレコードロック

この状態でユーザ B が顧客番号を入力しようとすると、まず最初に、制御データテーブルの中のキー値が 1 のレコードがリンクされるので、このレコードがロックされます。ところが、このレコードはすでにユーザ A がロックしているので、ロックの衝突が起こります。このために、ユーザ B はレコードロック解除待ちとな

ります（図 17-9）。

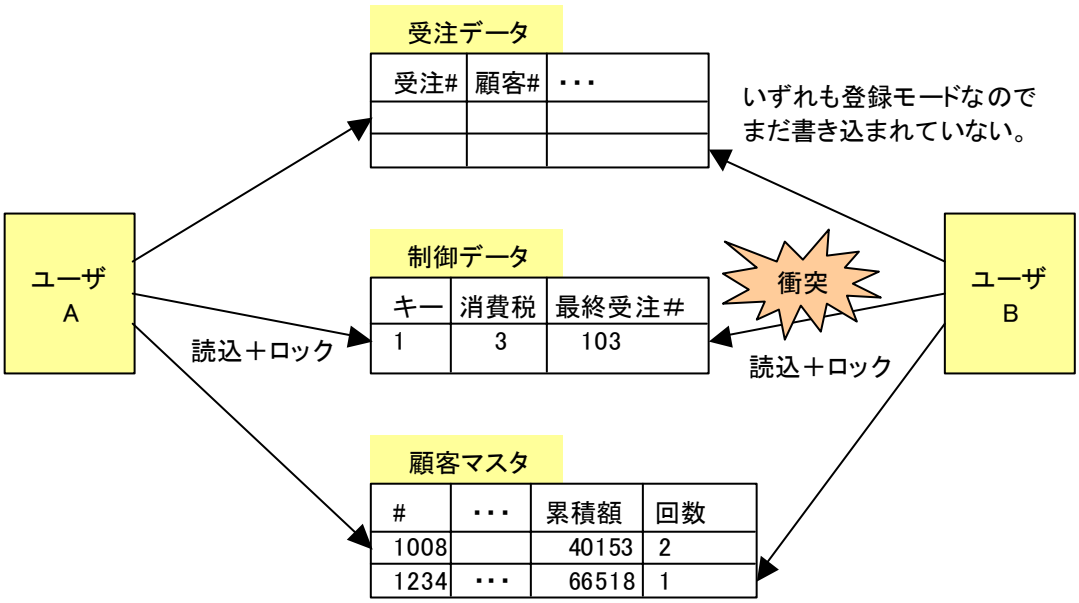


図 17-9 制御データのロック衝突

17.2.2. 対応

制御データには、レコードが 1 件しかなく、受注入力を行うプログラムはすべてこのレコードをリンクするので、このレコードに対するロック期間は極力短くしなければ、上に見たように、並行性に深刻な問題が起こります。

この問題に対応するため、次のようにプログラムを修正します。

ロックを回避するため、「アクセス」パラメータを「R=読込」にする

最初に、ユーザの入力中にはロックがかからないよう、DB テーブルで「アクセス」パラメータを「R=読込」に変更します。

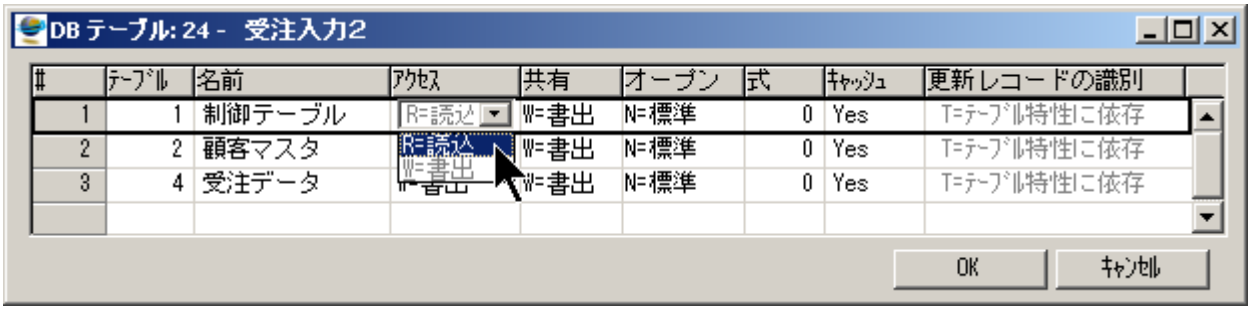


図 17-10 DB テーブルで、制御テーブルのアクセスを「R=読込」にしてロックを回避する。

項目更新を行わないようにする

ところで、この制御テーブルのレコードは、次の二つの目的のためにリンクされています。

- 消費税率と顧客へのメッセージを取得するため。これらのカラムへは参照のみでデータを変更することはありません。
- 登録モードのときに、新しい受注番号を発番するさいに計算の基準となる、最終受注番号を取得するため。

このカラムに対しては、発番を行った後に ＋1 します。

R=読込にしたら、このタスク内でデータを更新することができなくなりますので、後者を行えなくなってしまいますから、別のロジックで新しい受注番号を発番するようにしなければなりません。

簡単な方法として、新受注番号を発番するためのバッチタスクを作成します。このバッチタスクは制御データのレコードを読みこみ、最終受注番号の値を ＋1 し、その値をパラメータとして返す、というものです。このバッチタスクの実行中には、制御データのレコードロックがかかりますが、1 レコードだけを対象とするバッチタスクですから実行時間は極めて短く、ロックが衝突する確率は低くなります。

受注入力タスクでは、これまではレコード後処理で登録モードのときに (STAT(0,'C'MODE)=TRUE のとき) に、項目更新コマンドで最終受注番号を更新していましたが、今度はその代りに、新受注番号を発番するためのバッチタスクを呼び出すようにします。

受注番号発番バッチタスク

新受注番号を発番するバッチタスク「受注番号発番」は、図 17-11 のようなプログラムとなります。

- レコードメインで制御テーブルのレコードを呼び出し、レコード後処理で最終受注番号を＋1 します。そしてその値をパラメータとして返します。
- タスク終了条件が Yes であり、チェック時期が A=後置になっているので、このタスクは 1 レコードだけ処理を行った後終了します。

```
タスク特性：
  タスク名： 受注番号発番
  タスクタイプ： B=バッチ
  初期モード： M=修正
  タスク終了条件： Yes
  チェック時期： A=後置

レコードメイン：
  セレクト P=パラメータ P_受注番号　モデル： 受注番号
  リンク Q=照会 (1)制御テーブル インデックス：1
    セレクト R=実データ 1(制御キー)　範囲小： 1 範囲大： 1
    セレクト R=実データ 3(最終受注番号)
  リンク終了

レコード後処理
  項目更新 最終受注番号 式： 最終受注番号 ＋ 1
  項目更新 P_受注番号 式： 最終受注番号
```

図 17-11 受注番号発番バッチタスク[5]

[5] 以下、本書ではプログラムの説明のためにこのような書き方をしますが、Magic にこういったスクリプト言語があるわけではありません。

発番バッチプログラム呼び出しのタイミングは？

修正前の受注入力プログラムでは、登録時に新受注番号を計算するのはレコード前処理で項目更新コマンドで行っていました。

修正後に、新受注番号を発番するプログラムを同じ場所から呼び出すことも可能で、正しく動作します。ただし、こうした場合の問題点としては、受注入力プログラムを起動しただけでデータを入力せずに終了してしまった場合に、その受注番号は捨てられて、受注番号の歯抜けが起こってしまうことです。つまり、#103 の次が#105 になってしまって、#104 がない、というようなデータになる可能性があります。

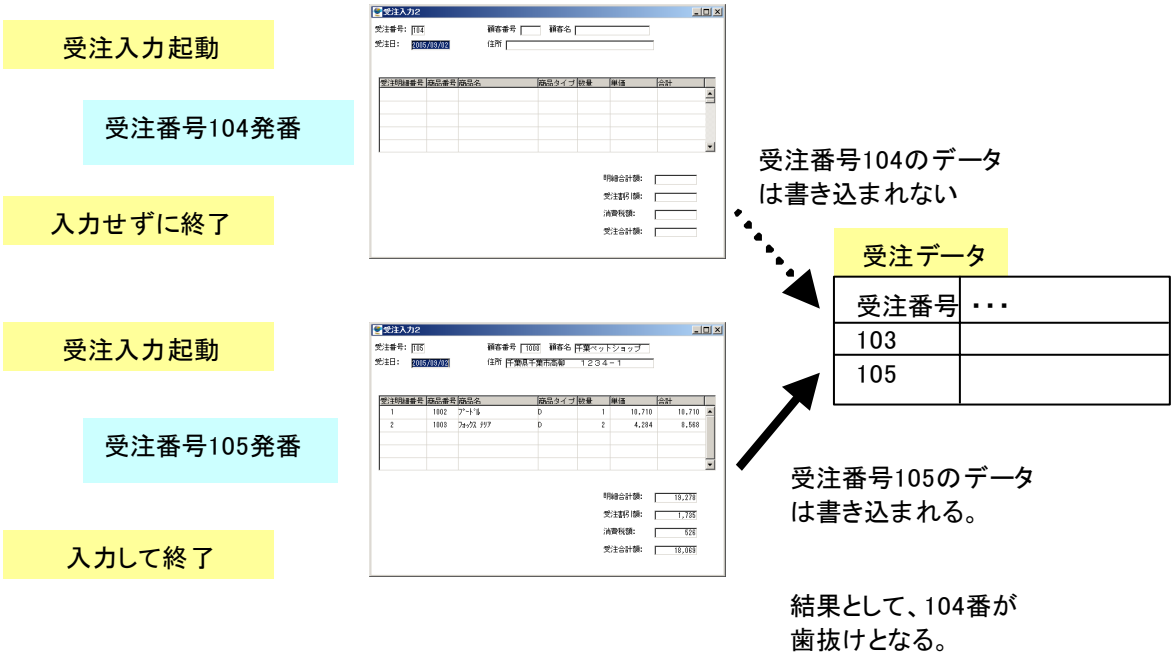


図 17-12 レコード前処理で発番すると、受注番号の歯抜けが起こる

これが許容できるかどうかは、システム(ユーザ)の要求仕様によります。もしこれでも良いのであれば、レコード前処理から呼び出して OK です。

しかし、もしできるだけ連番は歯抜けが起こらないようにしたい、という要求があれば、呼び出す場所(タイミング)を変えなければなりません。

では、レコード後処理で行えばよいでしょうか？レコード後処理はデータを実際に入力した場合にだけ実行されるので、歯抜けを防ぐためには OK ですが、問題は、子タスクで作成する明細レコードにも受注番号を設定しなければならないのですが、子タスクはレコードメインの実行中に呼び出されるので、レコード後処理では遅すぎる、ということです。

このため、受注番号は、レコードメインの実行中、子タスクを呼び出す前に発番しなければならないことになります。この例では、顧客番号を入力した直後くらいが一番適当であろうと思われます。これには「コントロール」レベルのハンドラを使います。

コントロール 後処理 のハンドラの定義

コントロールレベルのハンドラというのは、オンラインプログラムで特定のコントロールに関しての処理をまとめておくのに便利なハンドラで、「タスク定義」画面のレコードレベルの後に、以下のようにして定義します（図 17-13)。

1. タスクを開き、タスク定義画面で最下行(「R=レコード」「S=後処理」)にカーソルを置きます。

2. F4 キーで新しく行を作成する。→ レベルが「C=コントロール」、イベントが「P=前処理」の行が作成されます。
3. 「イベント」を「S=後処理」とします。
4. 「コントロール：」欄からズームすると、フォームに配置されているコントロールの一覧が表示されるので、この中から「顧客番号」を選択します。

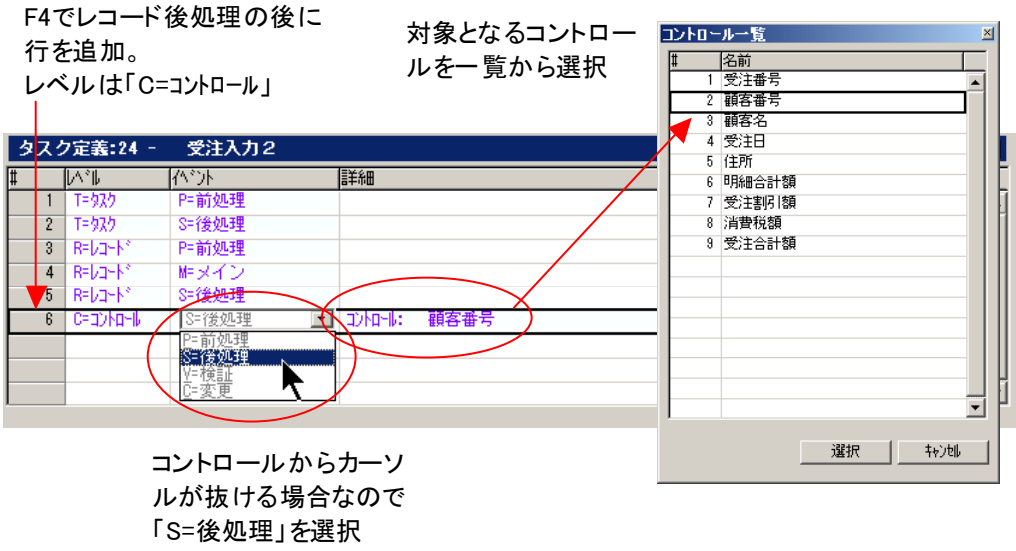


図 17-13 コントロール後処理レベルの追加

ここで、「レベル」が「C=コントロール」というのは、コントロールレベルのハンドラであることを意味します。

「イベント」欄は、このハンドラが実行されるタイミングを指定するもので、次のようなタイミングを設定できます。

イベント欄の設定	意味
P=前処理	コントロールにフォーカスが入ったタイミングで実行されます。
S=後処理	コントロールからフォーカスが外れたタイミングで実行されます。
V=検証	このコントロールの値に関する正当性の検証を行うための処理を定義します。このレベルは、このコントロールにフォーカスがない場合でも、他のコントロールの値が変わったりしたタイミングで実行されます。
C=変更	コントロールの値が、ユーザの入力により変更された場合に実行されます。

表 17-2 「イベント」欄の設定の意味

定義した処理レベルに対する処理内容は、「処理テーブル」で定義します。ここでは、「受注番号発番」タスクを呼び出します。このとき、パラメータとして受注番号を渡します。

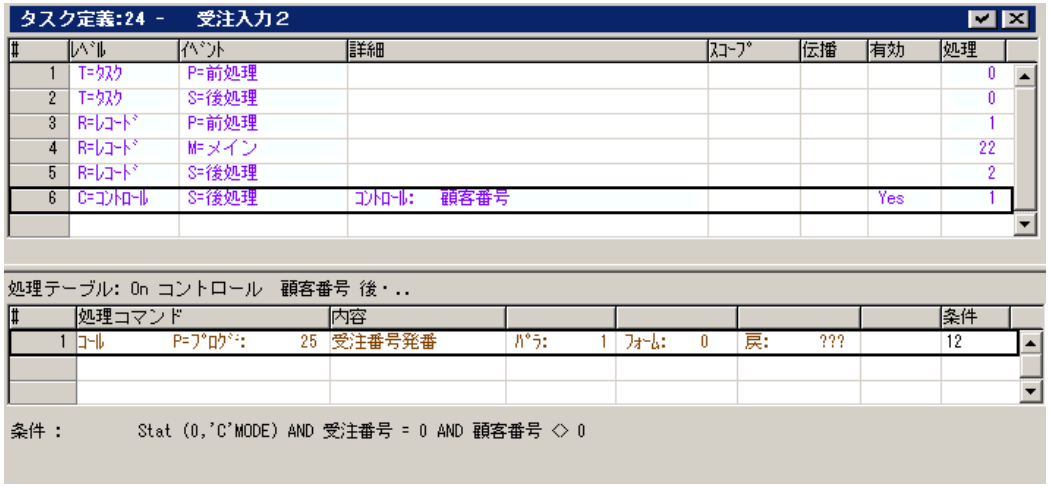


図 17-14 顧客番号のコントロール後処理で受注番号発番プログラムを呼び出す。

また、新しい受注番号は、登録モードで受注番号が未確定の場合にだけ呼び出すべきものなので、「条件」欄には、

```
Stat (0,'C'MODE) AND 受注番号 = 0 AND 顧客番号 <> 0
```

という条件を設定しておきます。

17.2.3. 実験

以上の変更をしてから、実際に実行して確認してみましょう。

1. 準備
   ① 開発版でアプリケーションを閉じます。
   ② 実行版を二つ立ち上げ、petshop アプリケーションを開きます。
   ③ 両方で、メニューから「受注入力２」プログラムを起動します。

この時点では、受注番号はまだ未確定なので、0 (表示上は空白)です。

2. 歯抜けが起こらないことの確認
   ① ユーザ A で、何も入力せずに ESC で終了します。
   ② 再度メニューから受注入力２プログラムを起動します。
   ③ 顧客番号#1008 を選択します。→ 受注番号が#104 に設定されます。

データの状態によっては、新しい受注番号が 104 以外になることもありますが、いずれにしても、何も入力せずに終了、ということをしたとしても、次に発番される受注番号は既存の受注データの最大値＋１になるはずです。

3. 制御テーブルのレコードロック解除待ちにならないことの確認
   ① ユーザ A が顧客 1008 を選択した状態で、ユーザ B の方で、顧客番号#1234 を選択します。

修正前のプログラムでは、ここで制御テーブルのロック解除待ちがでていましたが、今度は出ません。また、新しく受注番号が発番されますが、ユーザ A のものとは異なり、#105（あるいは、ユーザ A の番号＋１）に

なっているはずです。

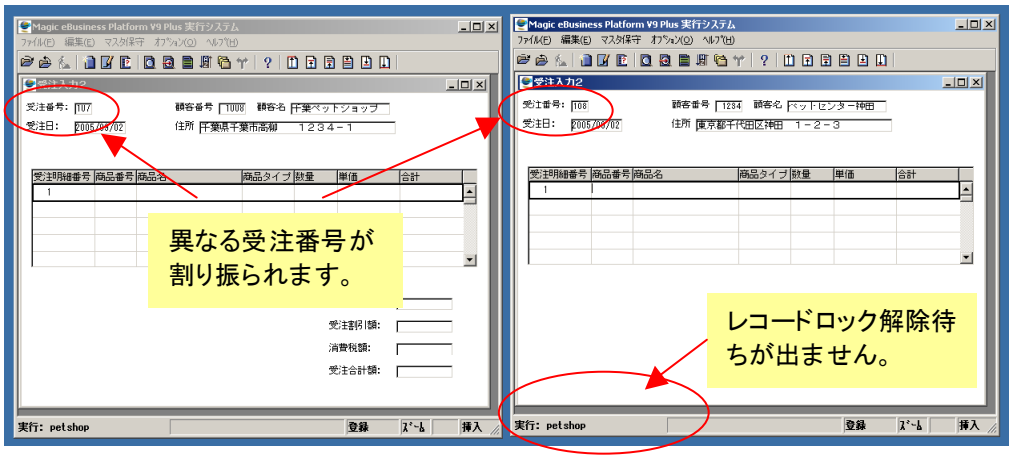


図 17-15 修正後のプログラムでは、制御テーブルのレコードロックの衝突が起こらない

このように、修正したプログラムでは、レコードロックの衝突も起こらず、また、連番も正しく発番されること、すなわち、歯抜けがなく、また、二つのユーザが同時に実行しても異なった番号が振られることが確認できました。

17.3. 顧客マスタのロック衝突

このようにして、制御テーブルのレコードロックの解除は回避することができましたが、問題はこれだけではありません。次に顧客レコードがロックの衝突を起こしてしまうケースを見ていきます。

17.3.1. 問題

今までの例では、ユーザ A は顧客番号 1008 を、ユーザ B は顧客番号 1234 を入力していました。ここで、ユーザ A、B ともに、同じ顧客番号 1008 を入力したら、何が起こるでしょうか？

1. ユーザ A、ユーザ B ともに、受注入力２プログラムを起動します。
2. ユーザ A で、顧客 1008 を選択します。
3. ユーザ B でも、顧客 1008 を選択します。

このようにすると、ユーザ B の方で「レコードロックの解除待ちです。テーブル： 顧客.MST」というエラーメッセージがステータス行に表示されます（図 17-16）。

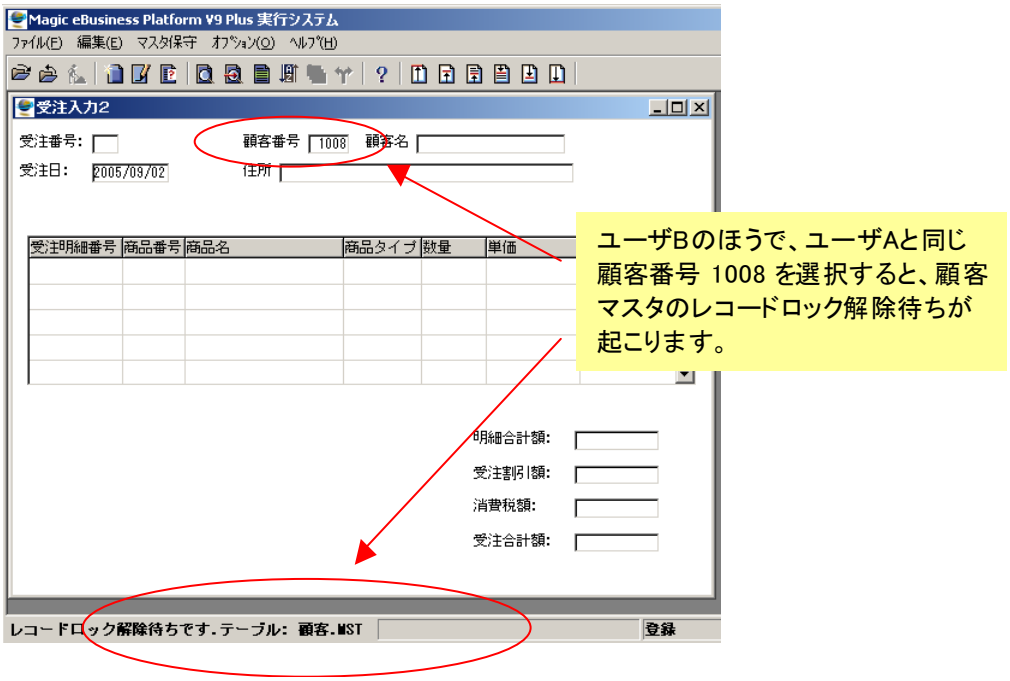


図 17-16 同じ顧客番号を選ぶと、顧客マスタでレコードロック解除待ちが起こる

なぜこうなるのでしょうか？

受注入力プログラムでは、制御レコードのほかに、顧客レコードもリンクしています。制御レコードは先ほどの修正で読込専用としレコードロックがかからないようにしたのですが、顧客レコードはそのままなので、レコードロックがかかります。ユーザ A が顧客 1008 を選択したときにこのレコードがロックされますから、ユーザ B も顧客 1008 を選択しようとすると衝突が起こります。このために「レコードロック解除待ち」のエラーが出るのです（図 17-17）。

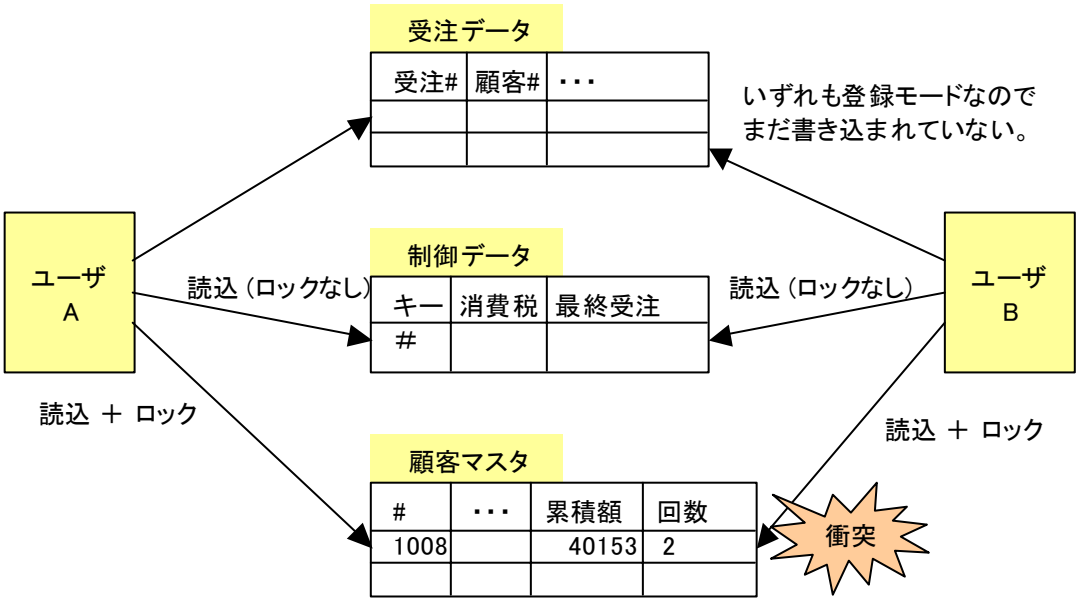


図 17-17 顧客レコードのロックの衝突

17.3.2. 対応

では、この問題にはどのように対応すればよいでしょうか？
一番簡単なのは、プログラムはこのままとし、こういう状況が起こらないように運用上カバーする、という方法です。顧客マスタのレコードロックの衝突が起こるのは、二人のユーザが同一顧客からの受注データを入力する場合にだけ起こります。しかし、データ入力担当オペレータの担当顧客を、例えば地域別などで決めておいたら、同一顧客からの受注データを別のオペレータが同時に入力する、ということはなくなります。従って、顧客レコードロック衝突の可能性はあるものの、運用上は問題ない、ということになります。
しかしながら、要求仕様上そうすることもできないときには、プログラムを対応させなければなりません。この場合のプログラムの修正方法は、基本的には、制御テーブルで行ったのと同じパターンです。すなわち、

- DB テーブルで「アクセス」パラメータを「R=読込」とし、レコードロックが発生しないようにする。
- レコード後処理などのレベルでデータを更新することができなくなるので、更新用のバッチタスクを作成し、このバッチタスクで更新させるようにする。

ただし、制御テーブルの場合には、バッチタスクでは単に最終受注番号を１増加させるだけなので非常に単純だったのですが、顧客マスタへのデータ修正はもう少し複雑です。受注入力プログラムでは、顧客マスタに対し親タスクのレコード後処理で次のような更新を行っています。

```
レコード後処理：
  項目更新 受注累計額　式： 受注合計額　計： I=加算
  項目更新 取引回数　式： 1　計： I=加算
```

ここでは、項目更新コマンドを「I=加算」モードで使っています。加算モードの項目更新コマンドは高度な機能を持っているので、プログラムの簡単化に非常に役立つのですが、顧客マスタのレコードをレコード後処理で更新できなくなってしまったので、残念ながらこれと等価なことを加算モードの項目更新コマンドを使わずに実行しなければならなくなってしまいます。

そこで、次に加算モードの項目更新コマンドの機能をまとめておきます。

17.3.3. 加算モードの項目更新の動き

加算モードの項目更新コマンドは、累計(注文額の累計、注文回数や個数の累計など)を格納した項目を更新するのに便利なコマンドです。通常のモードの項目更新コマンドでは、単純に式を計算しデータ項目に代入するだけなのですが、「加算」モードの場合には、レコードの初期状態を憶えていて、その初期状態で計算した式の値と、項目更新コマンドを実行した時点で計算した式の値との両方を計算し、その差分を、指定されたデータ項目に加える、という動作になります。

例えばレコード後処理の、次のような加算モードの項目更新コマンドについて考えて見ます。

```
項目更新　受注累計額　式：　受注合計額　計：　I=加算
```

このとき、受注番号#101 の受注データで、商品#1002 プードルの注文個数を 1 から 2 に変更した場合の計算を図 17-18 に示します。

① Magic エンジンは、レコードサイクルの最初で、レコードをフェッチした直後のビューの値を記憶しておきます (図中、「初期値」と書いてある、上半分の状態)。

② 加算モードの項目更新コマンドを実行するときには、

(1) まず、記憶しておいたフェッチ直後の値を元にして指定された式を計算します。このばあい、式は「受注合計額」そのものですので、初期状態での値 30,115 となります。

(2) 現在値をもとにして指定された式を再度計算します。これは、「現在値」と書かれている図中下半分の中の「受注合計額」になり、40,154 です。

(3) その差分を指定された項目に加算します。差分をとると +10,039 ですので、この値を、項目更新の対象となる「受注累計額」に加算すると、40,154 となります。

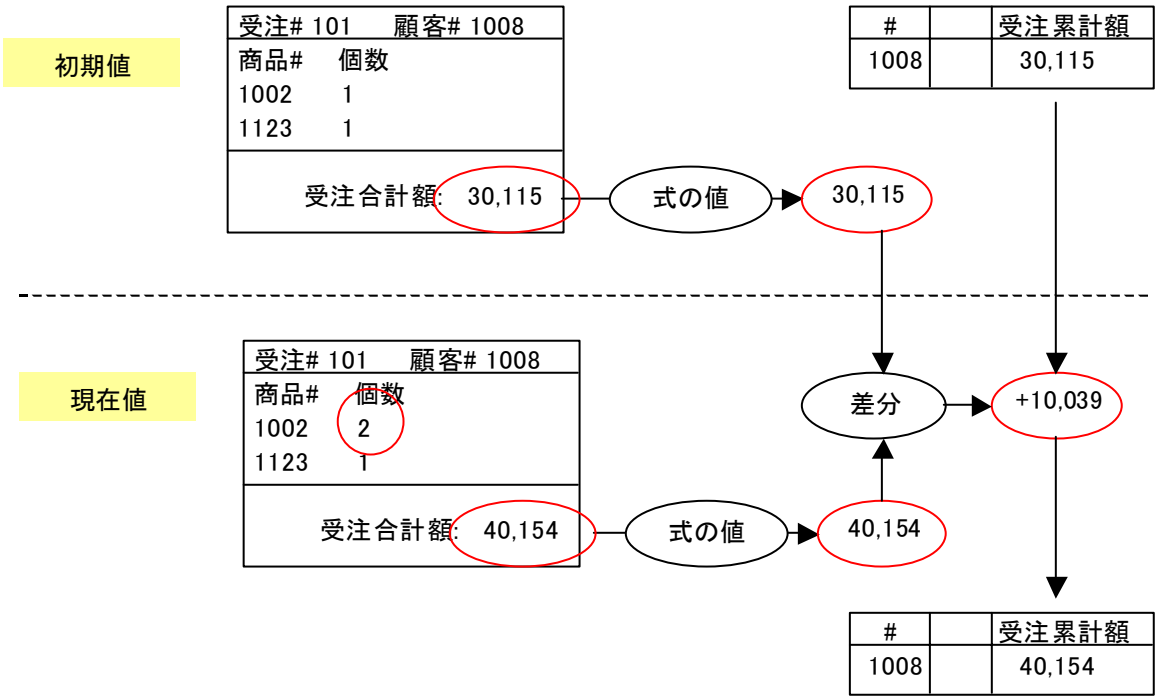


図 17-18 加算更新コマンドの計算

以上が基本動作ですが、登録モード、削除モードなどでは、次のように計算する必要があります。

- 登録モードの場合には、初期値は 0 とみなして計算する。すなわち、現在値がそのまま差分となります。登録モードということは、以前にはレコードが存在していないので、現在値がそのまま増加分となるからです。
- 削除モードの場合には、差分は「－現在値」とする。削除モードなのでこのレコードがなくなるのですから、現在値がそのまま減少分となるからです。

なお、レコード後処理の 2 番目にある加算モードの項目更新コマンドは、次のようになっています

```
項目更新 取引回数　式： 1　計： I=加算
```

ここで注意するのは、「式：」が定数の 1 になっていることですが、この項目更新コマンドがどういう効果を起こすのかというと、上の計算規則に照らし合わせてみると、

- 修正モードの場合には、式の計算結果は、初期値も現在値も 1 となり、差分は 0 となる。
- 登録モードの場合には、差分は現在値そのもの、すなわち 1 となる。
- 削除モードの場合には、差分はマイナス現在値、すなわち －1 となる。

となります。結果として取引回数は、修正するだけならば変更なく、登録モードでは＋1 となり、削除モードでは－1 となる、ということになります。

17.3.4. 差分更新プログラム

以上が、レコード後処理に定義されている加算モードの項目更新コマンドの効果ですが、話を元にもどすと、マルチユーザ環境では、顧客レコードを読込専用で読み込まなければならない関係上、レコード後処理で項目更新コマンドを実行できなくなるので、その代わりに、別のバッチタスクを立てて、そちらで顧客レコードの更新を行わせるようにする必要があります。

顧客レコード更新バッチタスク

顧客レコード更新用のバッチタスクは非常に単純です。図 17-19 に見るように、パラメータとして与えられる顧客番号で抽出した顧客レコードの受注累計額と取引回数を、同じくパラメータとして与えられるそれぞれの差分値で加算するだけです。

```
タスク特性：
  名前： 顧客マスタ差分更新バッチ
  タスクタイプ： バッチ
  初期モード： 修正
  タスク終了条件： Yes
  チェック時期： 後置
  メインテーブル： 顧客マスタ
  インデックス： 顧客番号

レコードメイン
  セレクト パラメータ P_顧客番号　モデル: 1（顧客番号）
  セレクト パラメータ P_累積受注額差分 モデル： 17(合計(10 桁))
  セレクト パラメータ P_取引回数差分 モデル： 15(注文/取引回数)
  セレクト 実データ 顧客番号　範囲小： P_顧客番号　範囲大： P_顧客番号
  セレクト 実データ 受注累計額
  セレクト 実データ 取引回数
```

```
レコード後処理
  項目更新 受注累計額　式： 受注累計額 + P_受注累計額差分
  項目更新 取引回数　式： 取引回数 + P_取引回数差分
```

図 17-19 顧客レコード更新バッチタスク

パラメータ

この顧客マスタ差分更新バッチタスクに与えるパラメータはどのようにしたらよいでしょうか？加算モードの項目更新コマンドの規則と同じことを、加算モードの項目更新コマンドを使わずにしたいのですから、登録モード、修正モード、削除モードに分けて考えると表 17-3 のようになります。

実行モード	受注累積額差分	取引回数差分
登録モード	受注合計額（現在値）	＋1
修正モード	受注合計額（現在値）　－受注合計額（初期値）	±0
削除モード	－ 現在値	－1

表 17-3 顧客マスタ差分更新バッチタスクに与えるパラメータ値

実行モードを Stat 関数で区別して、それぞれを Magic の式で書くと次のようになります。ここで、Q は受注合計額を表す変数記号です。

```
受注累積差分の式：
  IF (Stat (0,'C'MODE),Q,IF (Stat (0,'D'MODE),- (Q),Q-VarPrev ('Q'VAR)))
取引回数差分の式：
  IF (Stat (0,'C'MODE),1,IF (Stat (0,'D'MODE),- (1),0))
```

表 17-4 受注累積差分と取引回数差分の式

ここで、VarPrev という関数が出てきますが、これは変数の初期値を返します。VarPrev 関数へのパラメータは、変数そのもの(現在の変数の値ではなく)を VAR リテラルで渡します。上にあるように、変数 Q の初期値を得るには、VarPrev ('Q'VAR) とします。

最終的には・・・

以上のことをふまえ、親タスクのレコード後処理では、加算モードの項目更新の代わりに、顧客マスタ差分更新バッチタスクを呼び出すようにします。

(修正前) レコード後処理：

項目更新 受注累計額　式： 受注合計額　計： I=加算
項目更新 取引回数　式： 1　計： I=加算

(修正後) レコード後処理：

コール プログラム 顧客マスタ差分更新バッチ
　　パラメータ：

1. 顧客番号
2. IF (Stat (0,'C'MODE), 受注合計額,IF (Stat (0,'D'MODE),- (受注合計額), 受注合計額-VarPrev ('Q'VAR)))
3. IF (Stat (0,'C'MODE),1,IF (Stat (0,'D'MODE),- (1),0))

表 17-5 親タスクのレコード後処理の修正

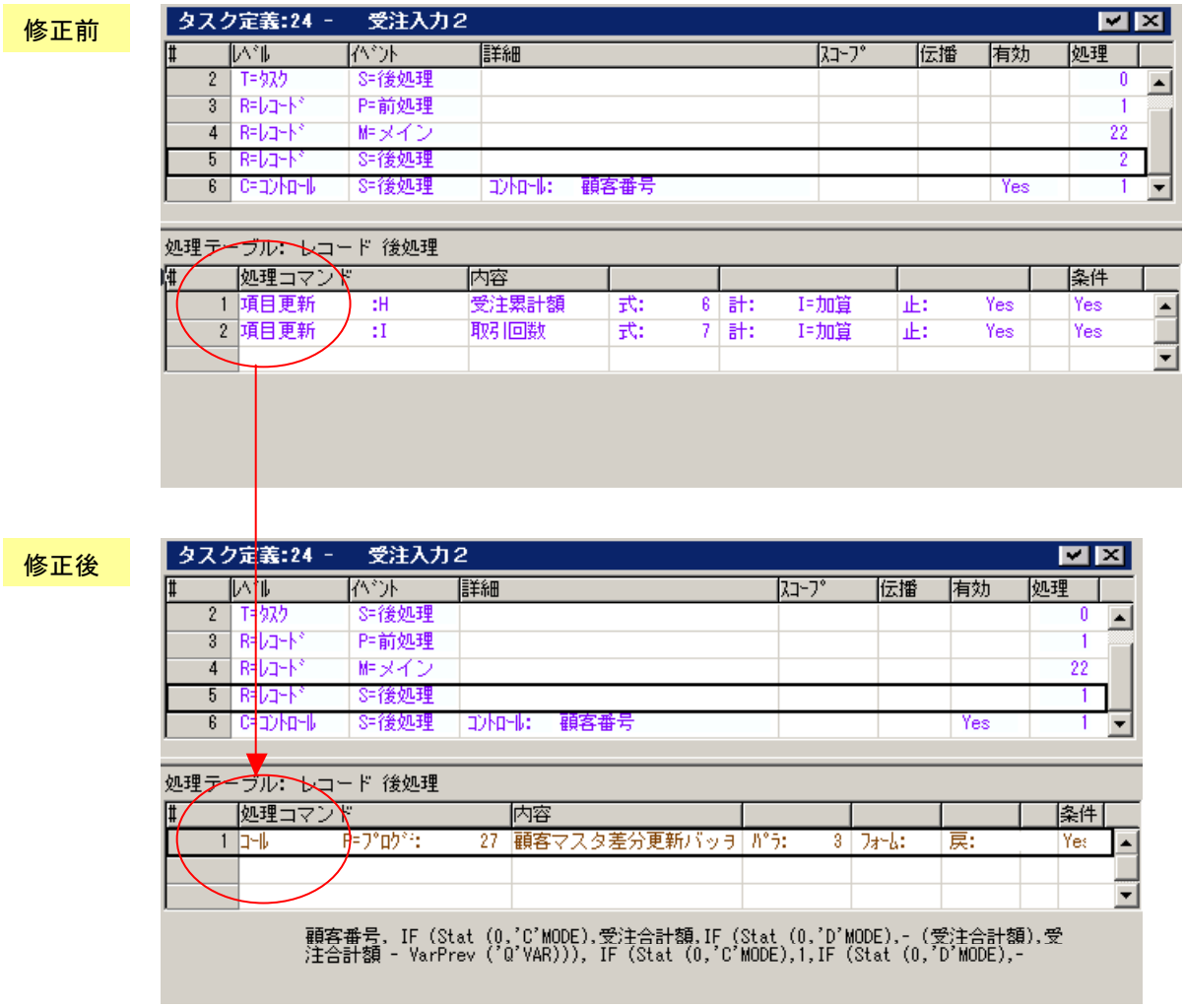


図 17-20 親タスクのレコード後処理の修正

17.3.5. 実行してみる

それでは、このように修正を施したプログラムを実行して、顧客レコードのレコードロックがかからないことを検証してみましょう。

1. 開発版でアプリケーションを閉じ、実行版を二つ起動し、それぞれアプリケーションをオープンします。
2. メニューより「受注入力２」プログラムを起動します。
3. ユーザ A で顧客#1008 を選択します。Tab キーを押すと、子タスクにカーソルが移動します。
4. ユーザ B で同じく顧客#1008 を選択します。以前のプログラムでは、この時点で顧客テーブルのレコ

ードロック解除待ちが出ていましたが、今度は出ません。

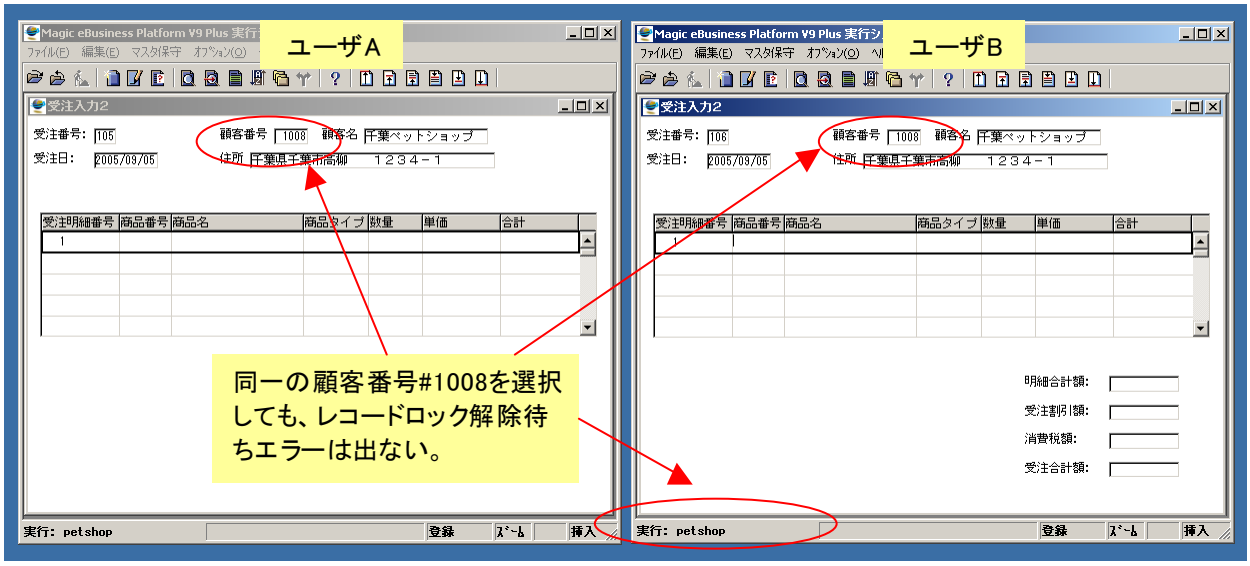


図 17-21 同一顧客番号を選択しても、レコードロック解除待ちが出ない

5. ユーザ A, B それぞれで、明細行に適当にデータを入力して、終了してください。
6. 受注入力２プログラムを終了し、メニューから「照会 – 顧客マスタ」プログラムを開いてください。
7. 受注累計額、取引回数が正しく更新されていることを確認してください。

顧客番号	顧客名	住所	割引率	条件	受注累計額	取引回数	顧客メモ
1008	千葉ペットショップ	千葉県千葉市高柳　１２３４－１	9.00	３０日後支払い	54,207	3	千葉ペット
1234	ペットセンター神田	東京都千代田区神田　１－２－３	10.00	現金	66,518	1	各種エサ、
3201	コジマペット	東京都足立区綾瀬　３－１１－５	5.00	現金			犬（自家繁
3220	ペットショップワンワン	東京都江戸川区南篠崎町　３－３２２	10.00	３０日後支			物・小
3321	ヤザキ金魚	東京都杉並区高井戸東　３－５－６	5.00	３０日後支			、通信
3440	ペットサロンヤザワ	東京都文京区小石川　２－５－７	3.00	現金			・ホテ
3550	ペットワールドハート	愛知県名古屋市名東区高針　３６５２	8.00	４５日後支払い	3,451	1	犬猫鳥から
4920	ANIMAL HOUSE	京都府京都市伏見区醍醐東大路町２３	5.00	現金			障害犬・猫
5133	山田ペットハウス	岡山県岡山市表町３－６－９	3.00	現金			ペット用品
5387	フクトミ鳥の友	佐賀県唐津市和多田本村５－５	5.00	現金			犬・猫・美
5493	わんわんペット	北海道札幌市厚別区厚別西３条２２－３	3.00	４５日後支払い			犬・美容・
5678	サンシャインペット	東京都豊島区池袋　４－１－１２	15.00	４５日後支払い			海水無脊椎

図 17-22 顧客マスタテーブルの受注累計額、取引回数

17.4. 商品マスタのロック衝突

今度は、子タスクの方を見ていきましょう。

17.4.1. ロック方式の変更

まず、マルチユーザに対応させるため、タスク特性のロック方式を変更する必要があります。子タスクのタスク特性から、「拡張(E)」タブを開いてください。図 17-23 のような設定になっているはずです。

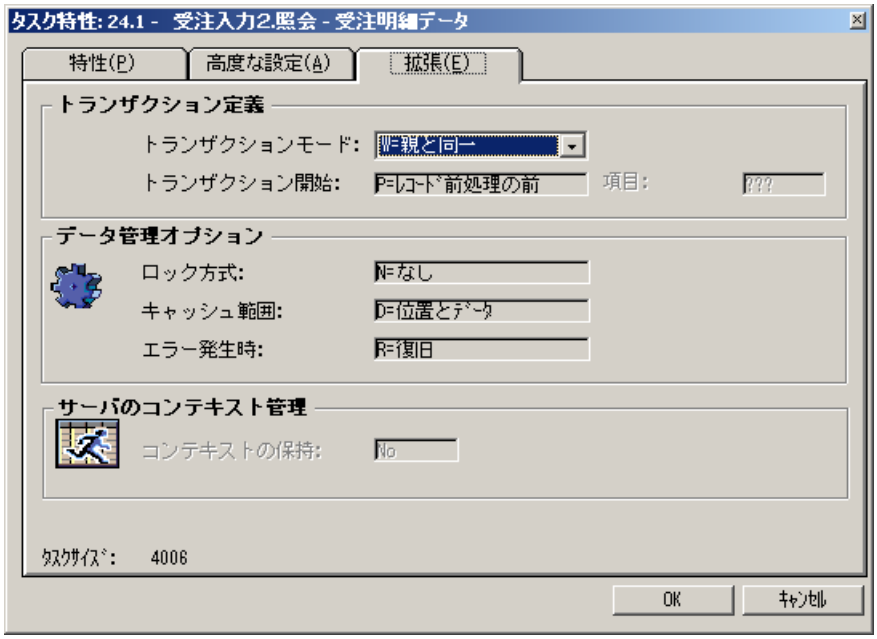


図 17-23 子タスクのタスク特性

ここで問題なのは、「ロック方式」が「N=なし」になっていることです。遅延トランザクションの設定の場合にはこれがデフォルトなのですが、今は遅延トランザクションを使わないので、このままではロックがかからず、更新の喪失の問題が起こる可能性があります。このため、「ロック方式」は「O=入力時」に変更する必要があります。

17.4.2. 商品レコードのロックの衝突

次に、次の手順で、明細行で同一の商品番号を選択してみてください。

1. ユーザ A、ユーザ B で、受注入力２プログラムをそれぞれ起動する。
2. 顧客をそれぞれ選択する。今回は同じ顧客でなくとも良い。
3. それぞれ明細行に進む。
4. ユーザ A で、商品#1002 (プードル)を選択し、Tab を押します。→ カーソルは、「数量」カラムに進みます。
5. ユーザ B で、同じく商品#1002 を選択します。→「レコードロック解除待ちです. テーブル: 商品.MST」というエラーがステータス行に表示されます。
6. ユーザ A で、個数を 1 と入力してから、↓矢印キーで次の明細行に移ります。→ ユーザ B の方で「レコードロック解除待ち・・・」が消えて、入力できるようになります。

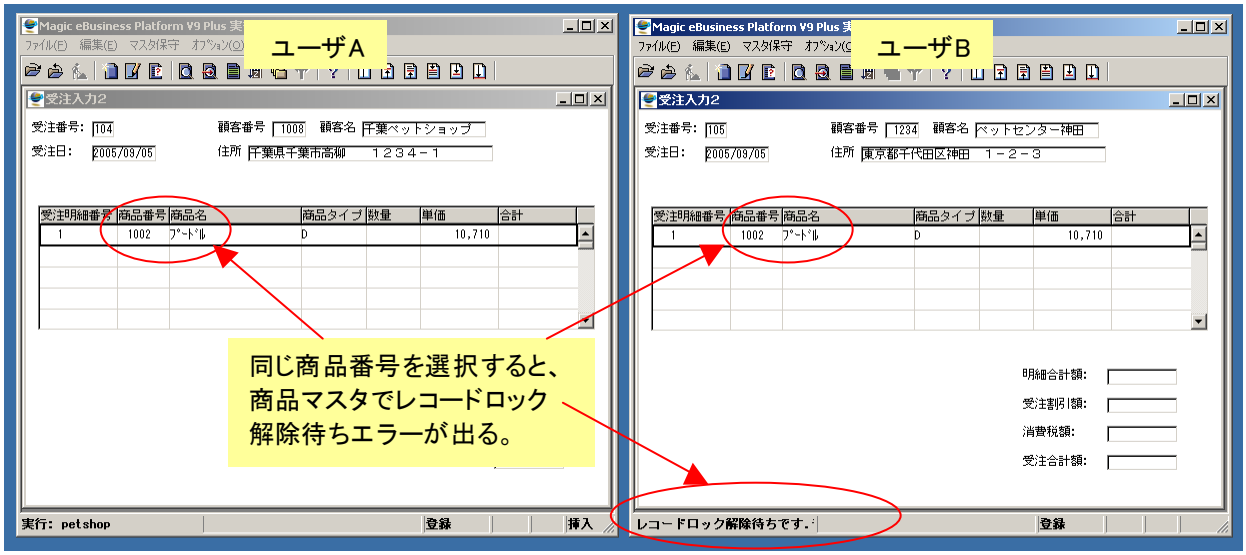


図 17-24 商品マスタでのレコードロック解除待ちエラー

これは、商品マスタのレコードでロックの衝突が起こっていることを意味します。これはどうして起こるのでしょうか？

理由は親レコードで顧客マスタのレコードでレコードロックの衝突が起きたのと同じです。すなわち、子タスクでは商品マスタのレコードをリンクしており、ユーザ A が商品番号を選択した瞬間に、この場合#1002 の商品レコードがリンクされるので、このレコードに同時のロックがかかることになります。ユーザ B も同じで、商品#1002 を選択したら、このレコードにロックをかけようとしますが、すでにユーザ A がロックをかけているのでロックの衝突となります。

17.4.3. 商品レコードのロック衝突への対応

このロックは、明細データで二人のユーザが同一商品を選択した場合に起こり、最初のユーザが次の明細行に移動した時に解除されます。そのため、ロックの衝突が起こったとしても、比較的短い時間で解除されるものです。親タスクの顧客マスタのレコードロックが、ひとつの受注データ入力が終わるまで解除されないのに比べると、待たされる時間が短く、影響も比較的小さいかもしれません。

しかし、それでも多くのオペレータが売れ筋商品を入力していたりすると、衝突で待たされる影響が大きく、運用上問題になるかもしれないので、対応方法を検討しておきましょう。

プログラムでの対応方法も、先に顧客マスタで行った方法と同じです。

1. 子タスクでは、商品マスタを読込専用とし、レコードロックがかからないようにしておく。
   これには、子タスクの DB テーブルを開き、「商品マスタ」の「アクセス」パラメータを「R=読込」に変更します。

2. 商品レコードを更新するバッチタスクを用意する。
   図 17-25 のような新しいバッチタスクを作成します。

```
タスク特性：
  名前： 商品マスタ差分更新バッチ
  タスクタイプ： バッチ
  初期モード： 修正
  タスク終了条件： Yes
  チェック時期： 後置
  メインテーブル： 商品マスタ
  インデックス： 商品番号

レコードメイン
  セレクト パラメータ P_商品番号　モデル: 6（商品番号）
  セレクト パラメータ P_受注数差分 モデル： 14(商品個数)
  セレクト 実データ 商品番号　範囲小： P_商品番号　範囲大： P_商品番号
  セレクト 実データ 受注数

レコード後処理
  項目更新受注数　 式：受注数+ P_受注数差分
```

図 17-25 商品マスタ差分更新バッチタスク

3. レコード後処理で、項目更新コマンドで商品レコードを更新する代わりに、このバッチタスクを呼び出してレコードの更新を行う。
   受注入力２の子タスクで、受注数を項目更新コマンドで修正する代わりに、新たに作成したバッチタスクを呼び出して更新します。

```
(修正前) レコード後処理：
  項目更新 受注数　式： 数量　計： I=加算
  ・・・

(修正後) レコード後処理：
  コール プログラム 商品マスタ差分更新バッチ
        パラメータ：
        1. 商品番号
        2. IF (Stat (0,'C'MODE), 数量,IF (Stat (0,'D'MODE),- (数量), 数量
           -VarPrev ('BA'VAR))
  ・・・
```

このように修正した後に、前と同じ手順で、受注明細行で同一商品を選択し、レコードロックの衝突が起こらないことを確認してください。また、受注数も正しく累積されること(更新の喪失が起こらないこと)も確認してください。

以上で、受注入力プログラムのマルチユーザ対応ができました。

18. 印刷

本章では、Magic の印刷機能について説明します。
印刷の基本は、テキストデータ出力の場合と基本的に同じで、タスクは通常次のようなつくりとなります。

- 照会モードのバッチタスク。
- 入出力ファイルテーブルで出力先(テキストデータ出力の場合にはファイル、印刷の場合にはプリンタ)を指定する。
- 出力用のフォームを定義する。
- データ出力コマンドで、必要なところでデータ出力を実行する。

印刷の場合には単純なテキスト出力とは異なり、ページという概念が入ってくるので、Magic でもページに関する便利な機能がいくつか取り入れられています。

参考情報： 本章の内容については、リファレンスマニュアルに詳しい情報がありますので参照してください。

プリンタテーブル	第 2 章 設定 → 設定／プリンタ
印刷プレビュー、ページヘッダとページフッタ	第 6 章 プログラム → 入出力ファイルテーブル → 入出力特性
フォームの定義	第 6 章 プログラム → フォームテーブル
フォントテーブル	第 2 章 設定 → 設定／フォント

18.1. 印刷の基本

18.1.1. APG で作成

まずは APG を使って、顧客一覧を印刷する簡単な印刷プログラムを作成して、プログラムの構造を見ていきましょう。

1. プログラムリポジトリを開き、最後に新しいプログラムを作成します。
2. Ctrl+G で、APG を起動します。→ APG ダイアログが表示されます（図 18-1）。
3. 「オプション」を「P=印刷」にします。
4. メインテーブルを 2 (顧客マスタ)にします。
5. OK ボタンを押すと、印刷プログラムが作成されます。

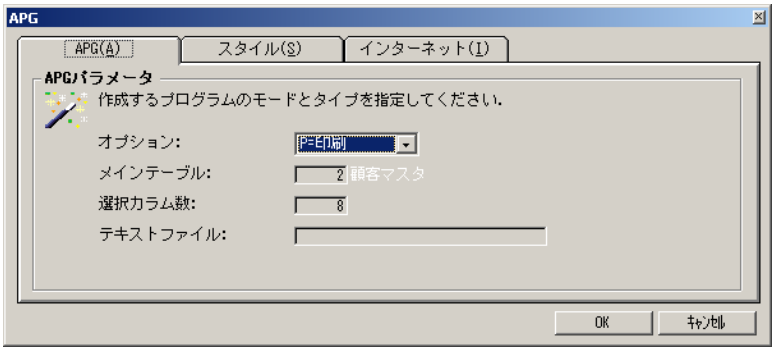
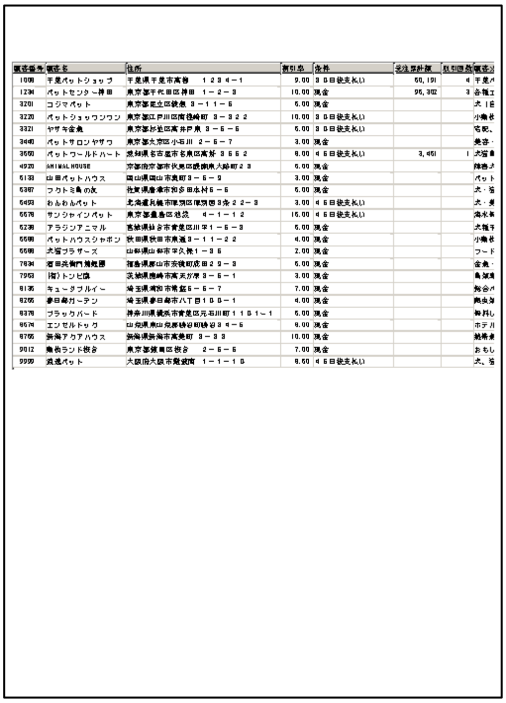


図 18-1 APG ダイアログ(印刷オプション)

作成したプログラムを実行してみましょう。F7 でプログラムを実行すると、図 18-2 のような出力が Windows でデフォルトとして設定されているプリンタで印刷されます。



顧客番号	顧客名	住所	割引率	条件	受注累計額	取引回数	顧客メ
1000	千葉ペットショップ	千葉県千葉市高洲 1234−1	9.00	30日後支払い	50,191	4	千葉ペ
1234	ペットセンター神田	東京都千代田区神田 1−2−3	10.00	現金	95,302	3	各種エ
3201	コジマペット	東京都足立区綾瀬 3−11−5	5.00	現金			犬 1日
3220	ペットショップワンワン	東京都江戸川区南篠崎町 3−322	10.00	30日後支払い			小動物
3321	ヤザキ金魚	東京都杉並区高井戸東 3−5−5	5.00	30日後支払い			宅配、
3440	ペットサロンヤザワ	東京都文京区小石川 2−5−7	3.00	現金			美容・
3550	ペットワールドハート	愛知県名古屋市名東区高針 3552	8.00	45日後支払い	3,451	1	犬猫用
4920	ANIMAL HOUSE	京都府京都市伏見区醍醐東大路町23	5.00	現金			雑種犬
5133	山田ペットハウス	岡山県岡山市奥町3−5−9	3.00	現金			ペット
5387	フクトミ鳥の友	佐賀県唐津市和多田本村5−5	5.00	現金			犬・猫
5493	わんわんペット	北海道札幌市厚別区厚別西3条22−3	3.00	45日後支払い			犬・美
5578	サンシャインペット	東京都豊島区池袋 4−1−12	15.00	45日後支払い			海水魚
6238	アラジンアニマル	宮城県仙台市青葉区川平1−5−3	5.00	現金			犬種
6598	ペットハウスシャボン	秋田県秋田市東通3−11−22	4.00	現金			小動物
6698	犬猫ブラザーズ	山梨県山梨市平久保1−35	2.00	現金			フード
7834	酒田奥衛門錦鯉園	福島県郡山市安積町成田29−3	5.00	現金			金魚・
7953	(有)トンビ屋	茨城県鹿嶋市高天が原3−5−1	3.00	現金			鳥類専
8135	キュータブルイー	埼玉県浦和市常盤5−5−7	7.00	現金			総合ペ
8265	春日部ガーデン	埼玉県春日部市八丁目100−1	4.00	現金			爬虫類
8378	ブラックバード	神奈川県横浜市青葉区元石川町1101−1	5.00	現金			無料し
8574	エンゼルドッグ	山形県東山形郡検沼町検沼34−5	8.00	現金			ホテル
8765	新潟アクアハウス	新潟県新潟市高美町 3−33	10.00	現金			熱帯魚
9012	動物ランド板谷	東京都板橋区板谷 2−5−5	7.00	現金			おもし
9999	浪速ペット	大阪府大阪市難波南 1−1−10	8.50	45日後支払い			犬、猫

図 18-2 顧客一覧印刷結果

では次に、このプログラムがどのように作られているかを、プログラムを開いて見ていきましょう。プログラムリポジトリからズームしてプログラムを開いてください。

18.1.2. 入出力ファイルテーブル

印刷出力を定義しているのは、入出力ファイルテーブルですので、最初にこれを見てみます。
入出力ファイルテーブルは、メニューからタスク環境(K) → 入出力ファイル(I)、あるいは Ctrl+I キーにより開くことができます（図 18-3)。

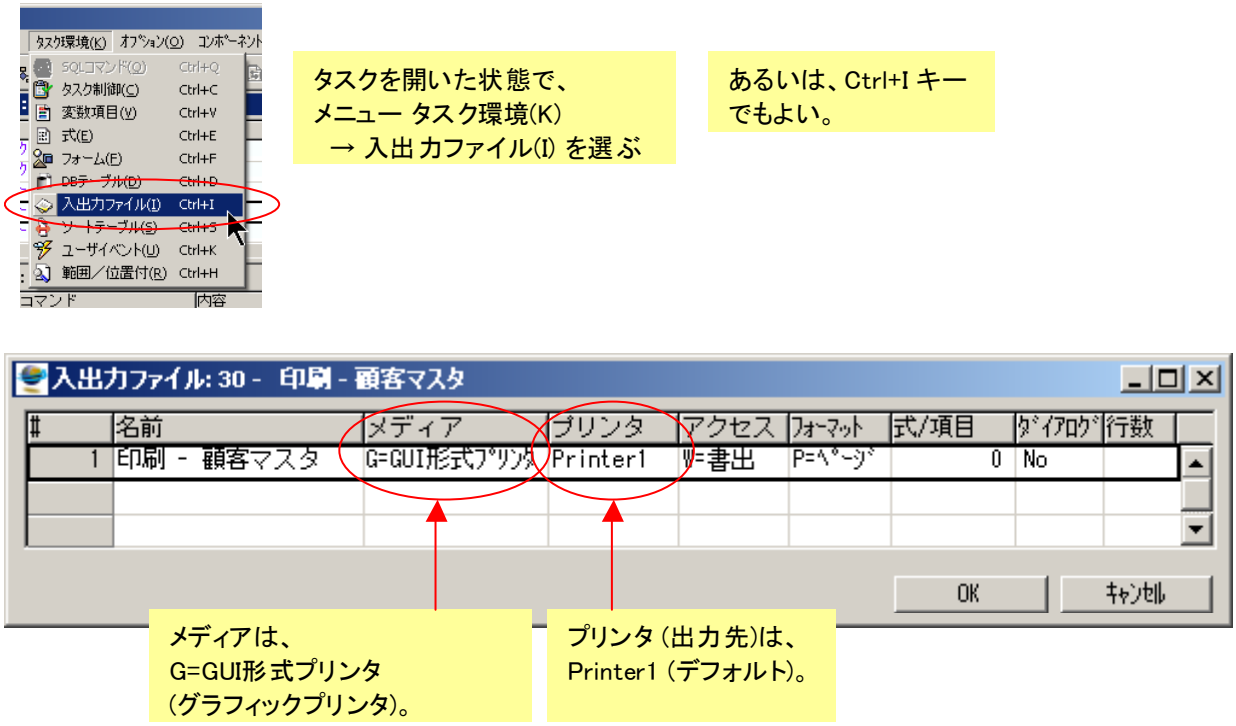


図 18-3 入出力ファイルテーブルで出力プリンタを定義

印刷出力の場合には、「メディア」は「G=GUI 形式プリンタ」(グラフィックプリンタ)、あるいは「P=テキスト形式プリンタ」を選びます。テキスト形式プリンタはテキストのみの印刷を高速に行いたい場合に使いますが、本章では GUI 形式プリンタのみを説明します。

Windows に複数のプリンタが定義されている場合には、どのプリンタに出力するかを指定する必要がありますが、これは「プリンタ」欄の設定で指定します。ここでは「Printer1」となっていますが、これは Magic のプリンタテーブルで定義されているプリンタの名前で、実際の Windows のプリンタとの関連付けはこのプリンタテーブルで行われます。詳しくは **18.2 プリンタの選択** で説明します。

18.1.3. フォーム

印刷結果の形式を定義するのは、出力用のフォームです。フォームはタスクのフォームテーブルで定義します。Ctrl+F でフォームテーブルを開くと、図 18-4 のようになっています。印刷出力用フォームは 3 番のものです。

- **クラス**は 1 となっています。クラス 0 のフォームはユーザとのオンラインのインタラクションを行うためのものですが、印刷出力の場合にはそれが不要なので、クラスは 1 となっています。
- **区分**は D=明細となっています。これはデフォルトの設定です。
- **インターフェースタイプ**は G=GUI 形式となっています。単純な文字だけでなくグラフィカルな出力を行う場合には、この設定にします。
- **名前**欄からズームすると、フォームエディッタに入り、フォームが表示されます。

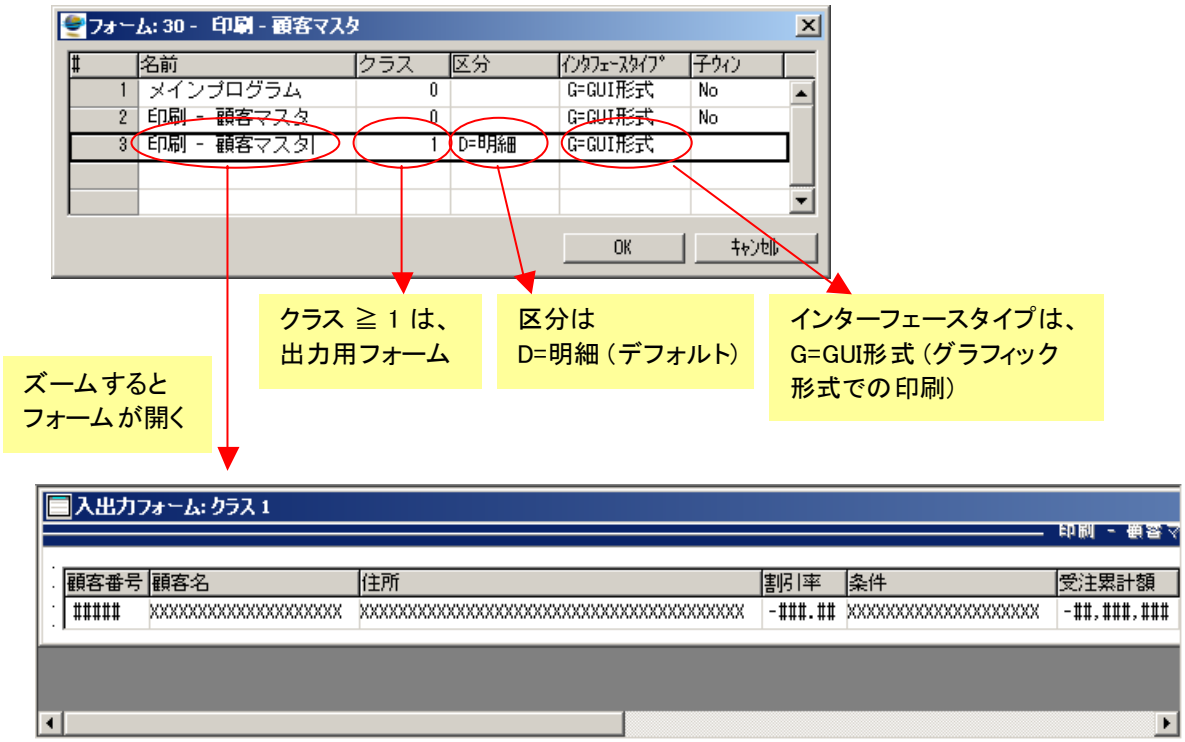


図 18-4 印刷用フォームの定義

フォームでは、テーブルコントロールが一つあり、全データ項目が配置されています。フォーム上ではテーブルコントロールには本体の行が 1 行しかありませんが、実際に印刷するとレコード件数分の行が印刷されます。また、レコード件数が多い場合には自動的に改ページして、複数ページに渡ってテーブルが印刷されるようになりますが、タイトル行は各ページごとに印刷されます[6]。

18.1.4. データ出力コマンド

このフォームを使い、実際のデータを印刷していくのは、**データ出力コマンド**です。
タスクでは 1 レコードに 1 行づつデータを印刷していくので、データ出力コマンドはレコード後処理で実行します（図 18-5）。
データ出力コマンドには、次のようなパラメータを設定します。

- 最初のパラメータは、出力時に利用するフォームの番号を指定します。番号を直接キー入力することもできますし、ズームして表示される一覧から選択することもできます。
- **ファイル**パラメータは、出力先を指定するもので、タスクの入出力ファイルテーブルの中の行番号を指定します。番号を直接キー入力することもできるし、ズームして表示される入出力ファイル一覧から選択することもできます。
- **頁**パラメータは、ページの最後まで印刷したらどうするかを指定するものです。デフォルトは A=自動で、これは自動的に改ページする設定です。

[6] テーブルコントロールの設定により、2 ページ目以降にタイトル行を印刷しないようにもできます。

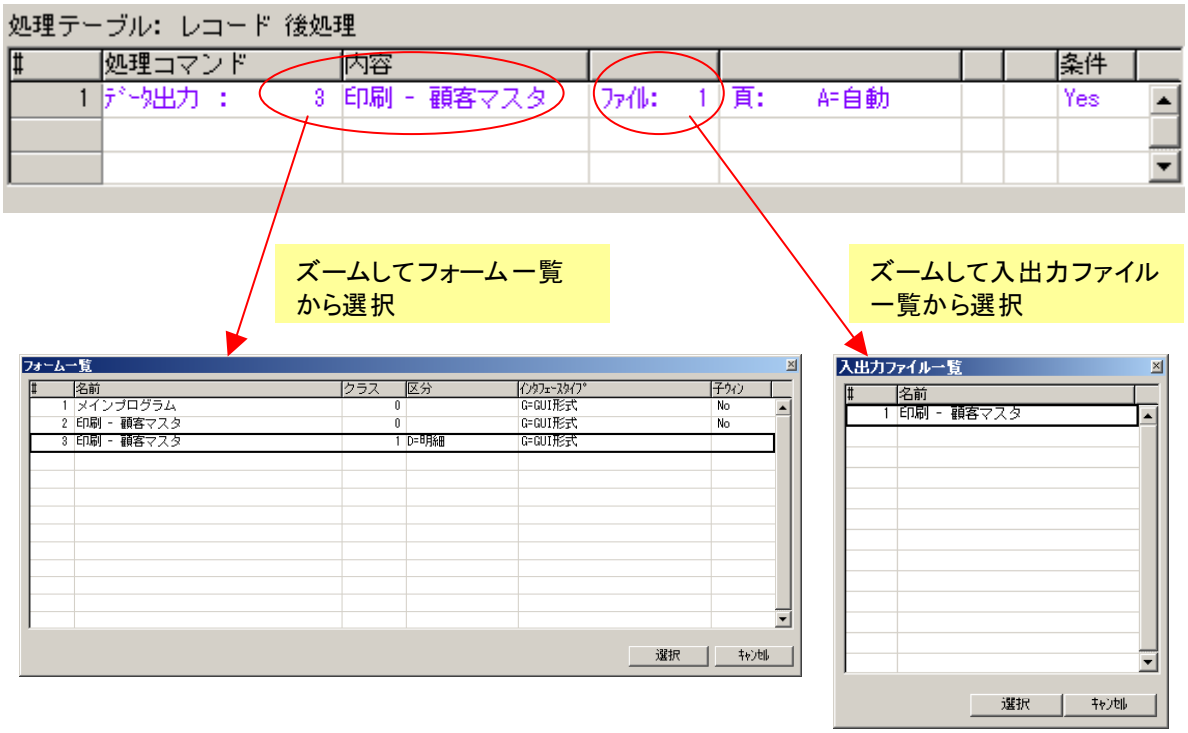


図 18-5 データ出力コマンド（レコード後処理）

以上のように Magic プログラムでの印刷は、① 入出力ファイルテーブル、② フォーム、③ データ出力コマンドが基本です。

18.2. プリンタの選択

デフォルト以外のプリンタに印刷したい場合にはどのような設定をするのでしょうか？ Magic のプリンタ選択の方法を理解するには、まず Magic の**プリンタテーブル**について理解する必要があります。

18.2.1. プリンタテーブル

プリンタテーブルとは、Magic が管理する設定情報のひとつで、Windows 上に定義されているプリンタと、Magic のプログラムで指定されているプリンタとのマッピングを取っています。プリンタテーブルは MAGIC.INI ファイルに定義される設定情報なので、環境により Windows プリンタの名前などが異なっても、プログラムを変更する必要はなく、MAGIC.INI での記述のみを変更すればよいようになっています。
プリンタテーブルは、アプリケーションを閉じた状態で、メニュー 設定(S) → プリンタ(P) を選択するか、あるいはツールバーから開くことができます（図 18-6）。

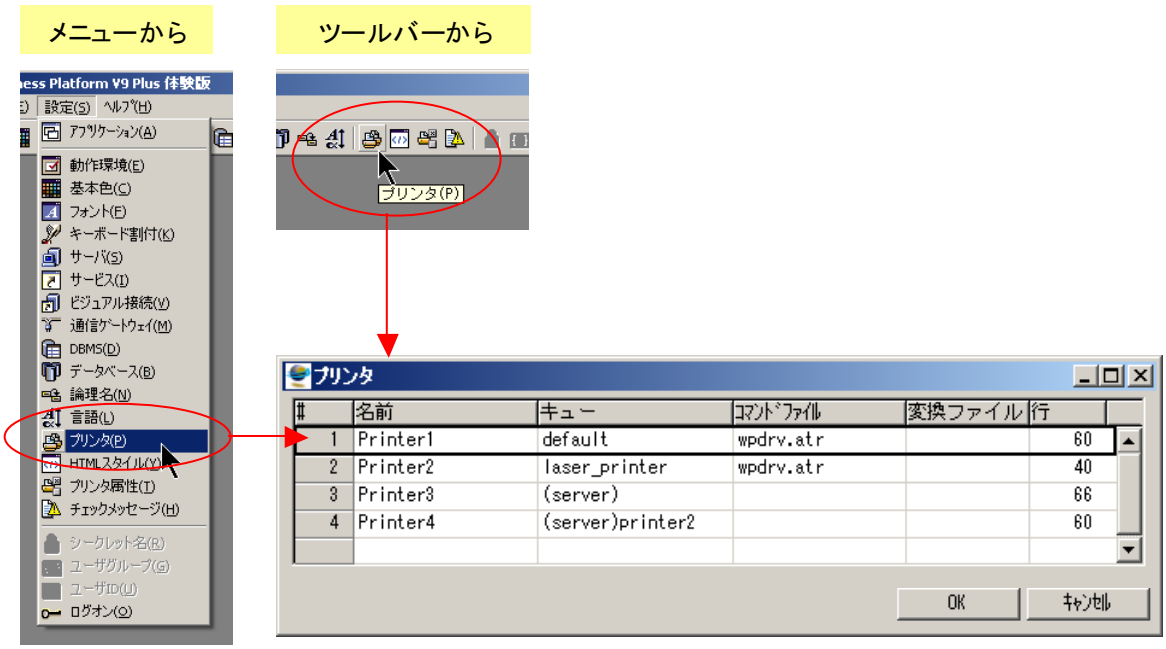


図 18-6 プリンタテーブル

プリンタテーブルには、次のような欄があります。

- **名前**： Magic プログラムが参照する名前です。具体的には、タスクの入出力ファイルテーブルの「プリンタ」欄で参照します。この名前は Magic の中でだけ有効なもので、日本語を含む任意の文字列を使うことができます[7]。
- **キュー**： Windows に定義されているプリンタの名前を指定します。もしこの名前に一致するプリンタが Windows 上で定義されていなければ、デフォルトのプリンタに出力します。
- **コマンドファイル**： テキスト形式プリンタの場合に利用するものですが、ここでは説明を省略します。
- **変換ファイル**： これもテキスト形式プリンタの場合に利用しますが、これも省略します。
- **行**： 1 ページに印刷できる行数を指定します。

[7] カンマ文字「,」など、使えない特殊文字が若干ありますので、特殊文字はなるべく避けてください。

18.2.2. 実行時のプリンタの選定

Magic が実行時に出力するプリンタは、図 18-7 のような順序で決定されます。

1. データ出力コマンドの「ファイル」欄の設定で、タスクの入出力ファイルテーブルに定義されている入出力ファイルの一つを選択します。(例では#1 の「印刷 – 顧客マスタ」)
2. 入出力ファイルテーブルでは、「プリンタ」欄の設定から、同名のプリンタ名をプリンタテーブルの中から探します。例では、「Printer1」です。
3. プリンターテーブルで、選択されたプリンタ Printer1 の「キュー」欄を見ます。ここに指定されている名前と同じ名前を持つ Windows プリンタを探します。もし見つかればそこに出力されますが、もし同名のプリンタがなければデフォルトプリンタに出力されます。

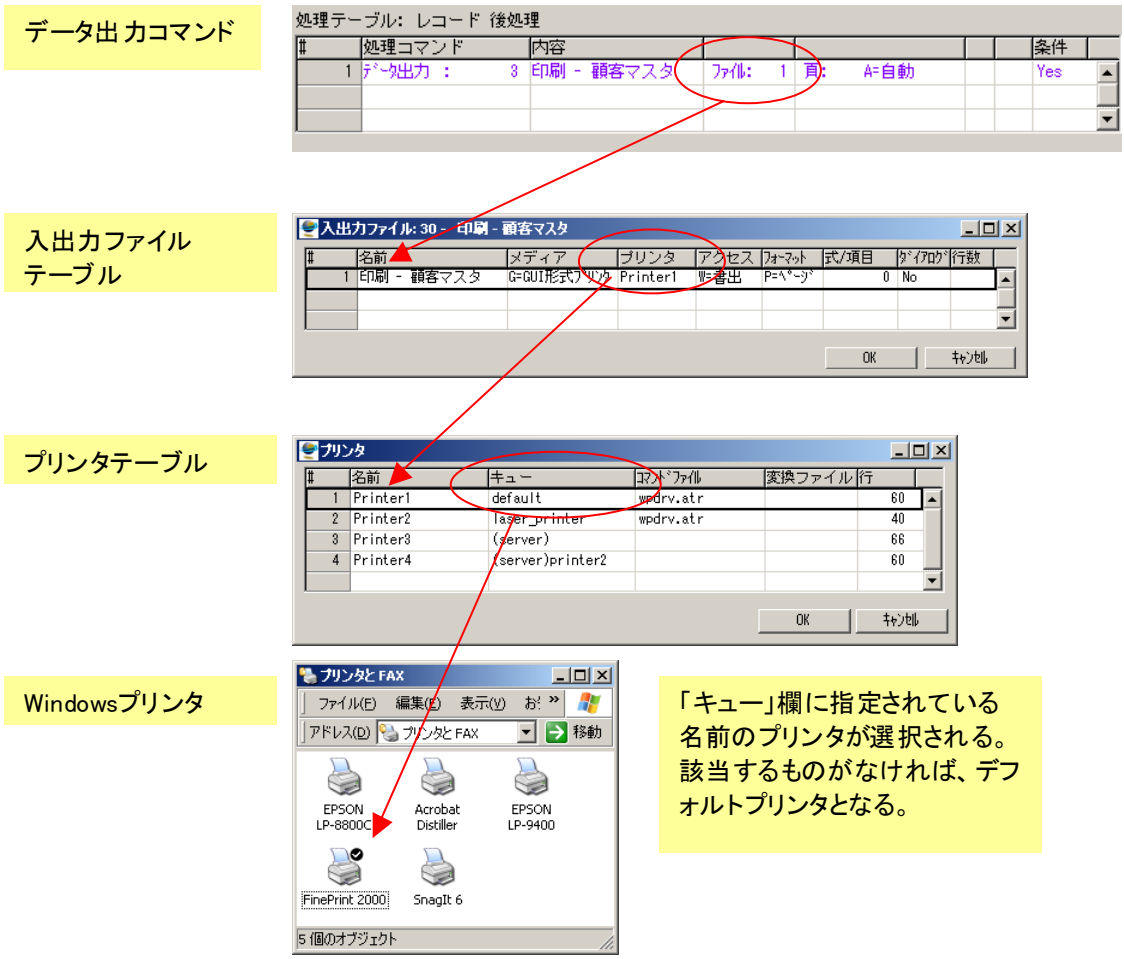


図 18-7 プリンタの選択

18.2.3. 印刷出力先の変更

以上のことから、印刷出力先を変えるには、① データ出力コマンドで「ファイル」パラメータを変える、② 入出力ファイルテーブルで、「プリンタ」設定を変える、③ プリンタテーブルで「キュー」欄を変える、という方法があることがわかります。

このうち、①および②の方法はプログラムの変更であり、③は MAGIC.INI の変更となります。利用目的にあわせて、いずれかの方法を選択してください。

18.3. 印刷プレビュー

Magic は簡単な印刷プレビューの機能を備えています。

デフォルトではプレビューがされませんが、プレビューを行わせるには、**入出力特性**ダイアログで指定します（図 18-8）。

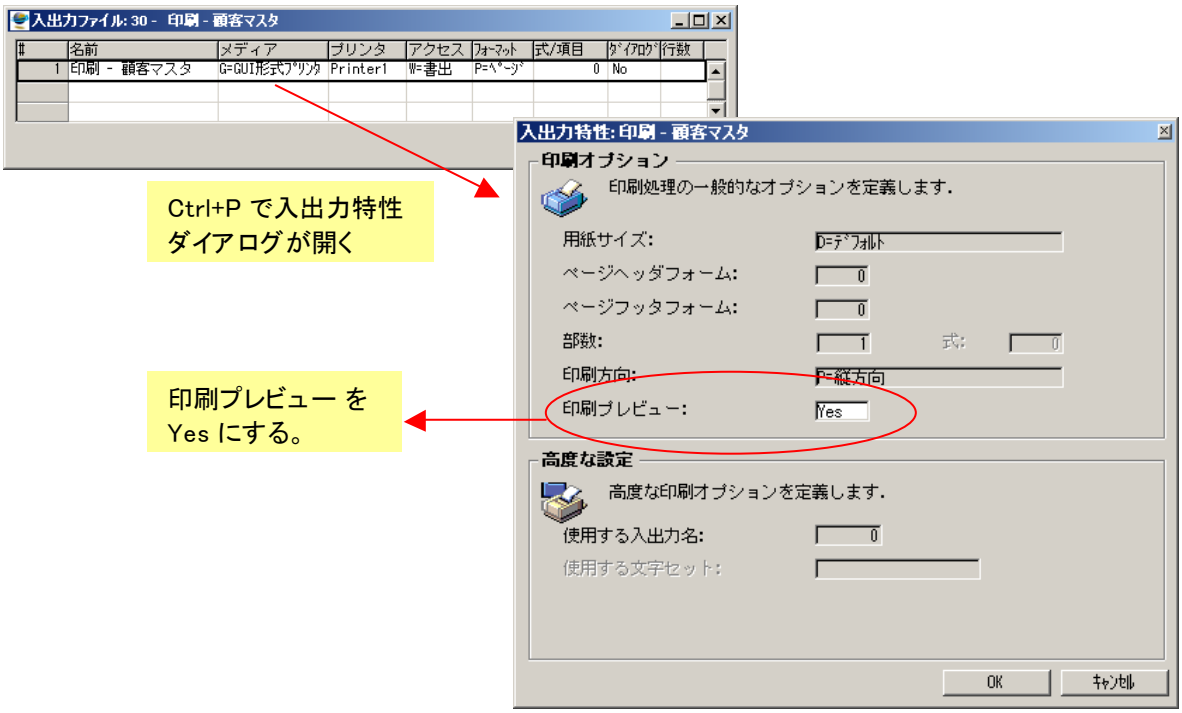


図 18-8 印刷プレビュー パラメータ（入出力特性ダイアログ）

設定の方法は次の通りです。

1. タスクを開いた状態で、Ctrl+I で入出力ファイルテーブルを開く。
2. プレビューを行わせたいプリンタにカーソルを置いて、Ctrl+P キーを押す → 入出力特性ダイアログが開く。
3. **印刷プレビュー**パラメータを Yes にする。

これで、実行時に印刷出力がプレビュー画面（図 18-9）に表示されます。ここからキャンセルすることもできるし、実際に印刷を行わせることもできます。

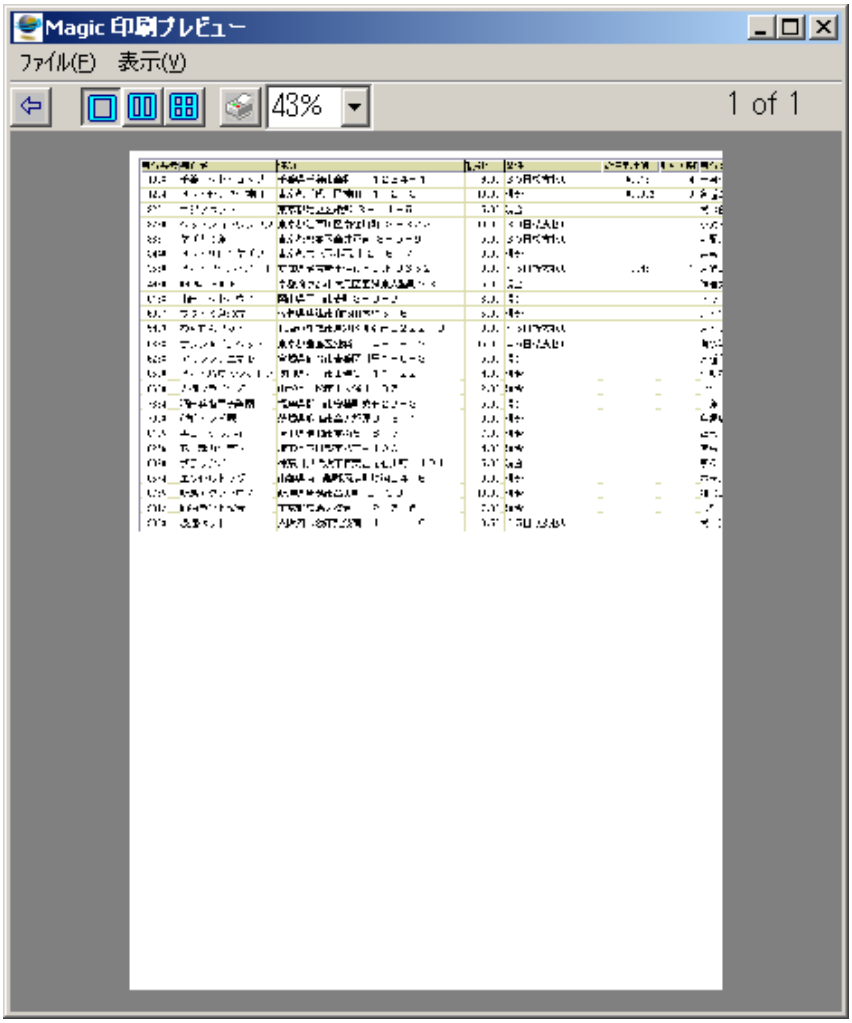


図 18-9 プレビューの出力結果

18.4. ページヘッダとページフッタ

帳票印刷では、データだけをべたべた並べるだけでなく、ページの体裁を整えることも重要です。その中でもページヘッダとページフッタは基本です。

ページヘッダとページフッタの例を図 18-10 に示します。ページヘッダ/フッタは、各ページの先頭および下端に印刷されるもので、ドキュメントの題目、日付、時間、ページ数、会社名、「社外秘」などの注釈等が、全ページに印刷されます。図 18-10 では、ヘッダに題目と日付、時間が、ページフッタにはページ数が印刷されています。

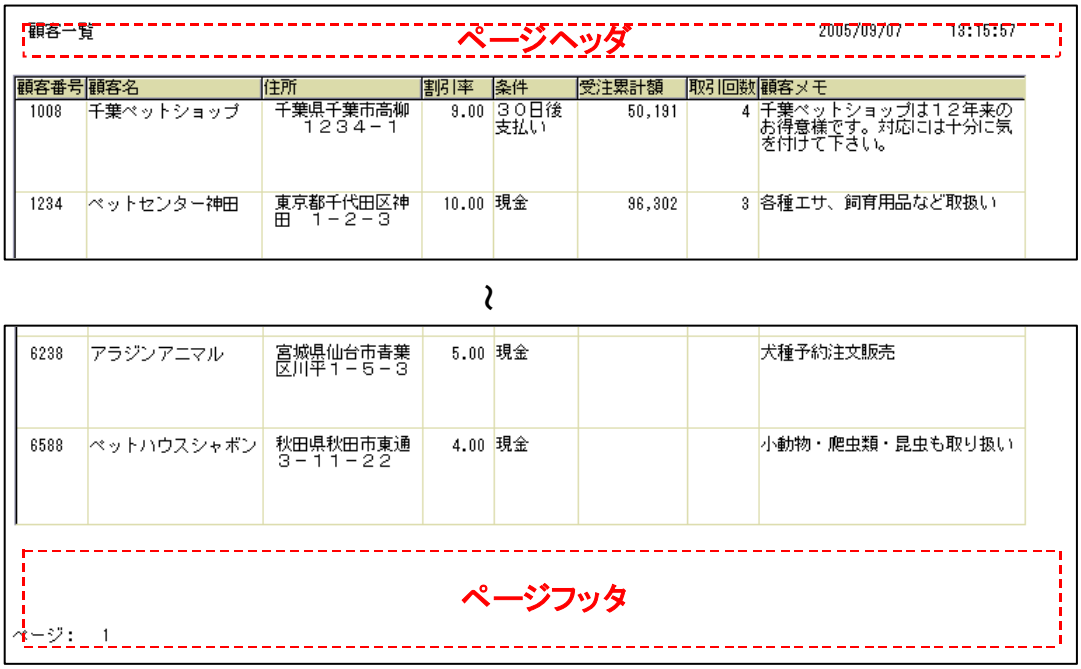


図 18-10 ページヘッダとページフッタ

Magic のプログラムでもページヘッダとページフッタとを簡単に印刷する機能が備えられていて、ページヘッダ/フッタ用のフォームの定義と、入出力特性での設定、という手順で定義します。

18.4.1. フォームの定義

ページヘッダ/フッタは、タスクのフォームテーブルでそれぞれ専用のフォームを定義します。図 18-11 は、前に APG で作成した印刷プログラムに、ページヘッダ/フッタのフォームを追加したものです。これは次のようにして定義します。

1. 最初は 区分 欄が D=明細 のフォームだけがありましたが、その前後にクラス 1 のフォームをひとつづつ作成し、区分 をそれぞれ P=ページヘッダ、G=ページフッタ とします。
2. ページヘッダ、あるいはページフッタのフォームにカーソルを合わせ、フォーム特性の「高さ」特性をそれぞれ 1 にします。これは、ページヘッダ、ページフッタの高さを 1cm にする、という意味です。
3. 名前欄からズームすると、フォームエディッタが開き、フォームが表示されます。このとき、同一クラス(クラス=1)のフォームがすべて一緒に開かれます （図 18-12)。
4. ページヘッダのフォームには題目「顧客一覧」をテキストコントロールとして配置します。

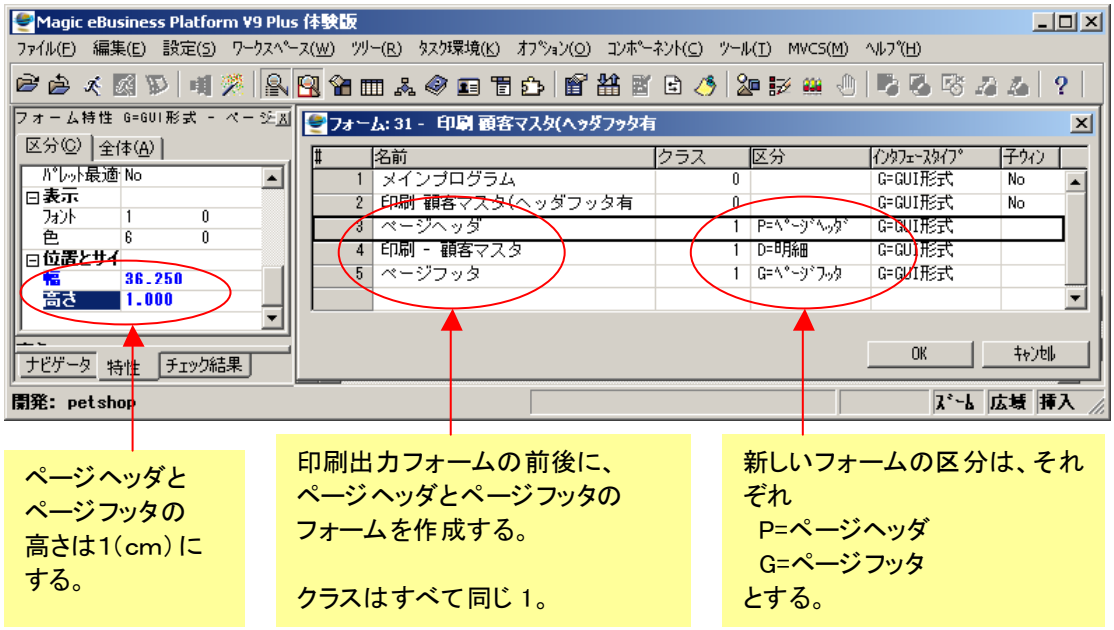


図 18-11 ページヘッダとページフッタの作成

5. 日付を表示させるために、コントロールパレットからエディットコントロールをページヘッダフォームに配置します。このエディットコントロールをマウスで選ぶと、特性シートには「コントロール特性： エディット」としてエディットコントロールの特性が表示されます。この中で「データ」の行の右側の欄からズームすると、式テーブルが開きますので、ここで Date() と指定します。これは今日の日付を返す日付型の組み込み関数です。

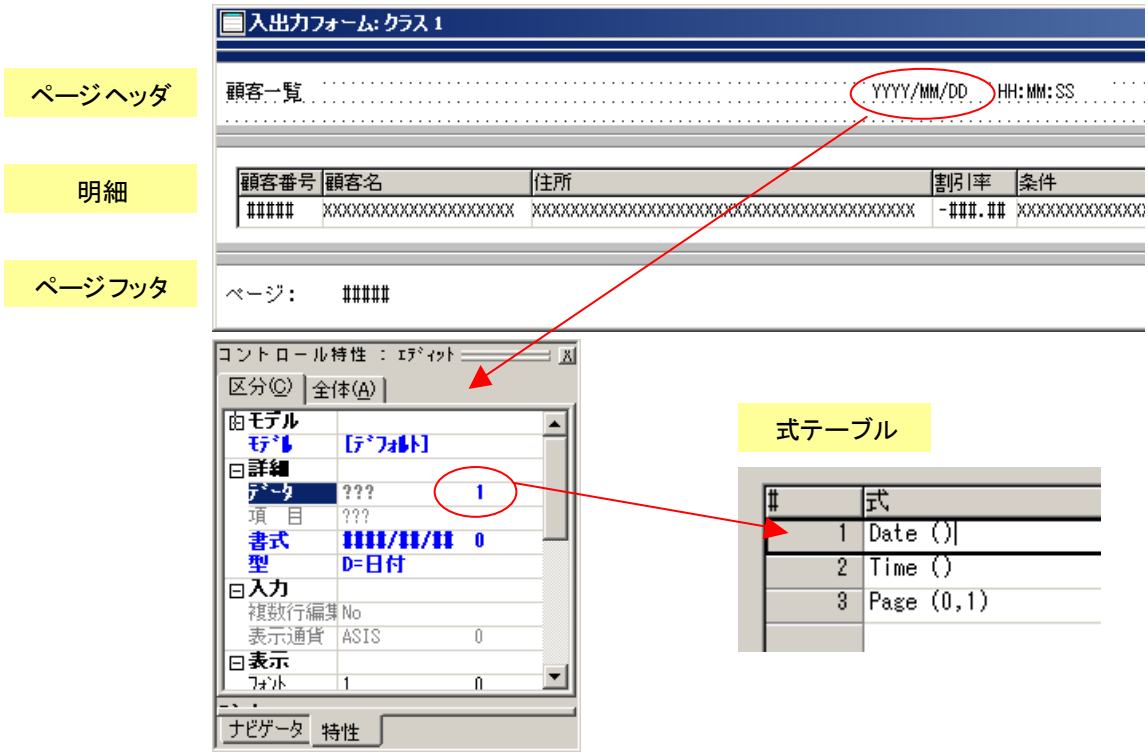


図 18-12 フォームエディッタでのフォーム設計

6. 同様にして、ページヘッダに時刻、ページフッタにページ数を表示するように、コントロールを配置して

ください。

参考： 今日の日にちは Date、現在の時刻は Time、ページ数のカウントには Page 関数を使います。Page 関数のパラメータには、タスクの世代(自タスクが 0、親タスクが 1、・・・)と、入出力ファイルの番号(入出力ファイルテーブル中での行番号)を指定します。

以上で、フォームの作成は終了です。

18.4.2. 入出力特性の設定

ページヘッダ/フッタフォームを定義すれば、あとはそれを入出力特性に設定するだけです。

入出力特性は、入出力ファイルテーブルを開き、プリンタの行にカーソルを置き、Ctrl+P キーを押すと表示されます。この中の「ページヘッダフォーム」「ページフッタフォーム」欄に、ページヘッダ/フッタのフォーム番号を指定します（図 18-13)。フォーム番号を直接キー入力することもできますし、ズームするとフォームの一覧が表示されるので、ここから選んでも構いません。

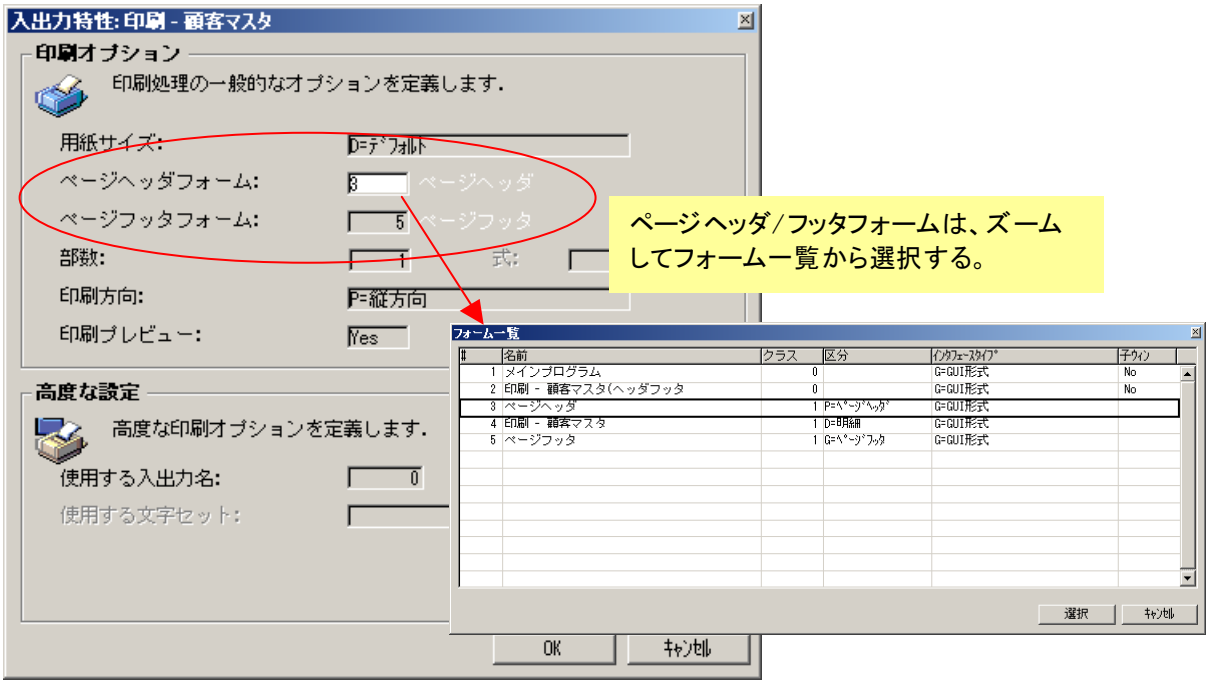
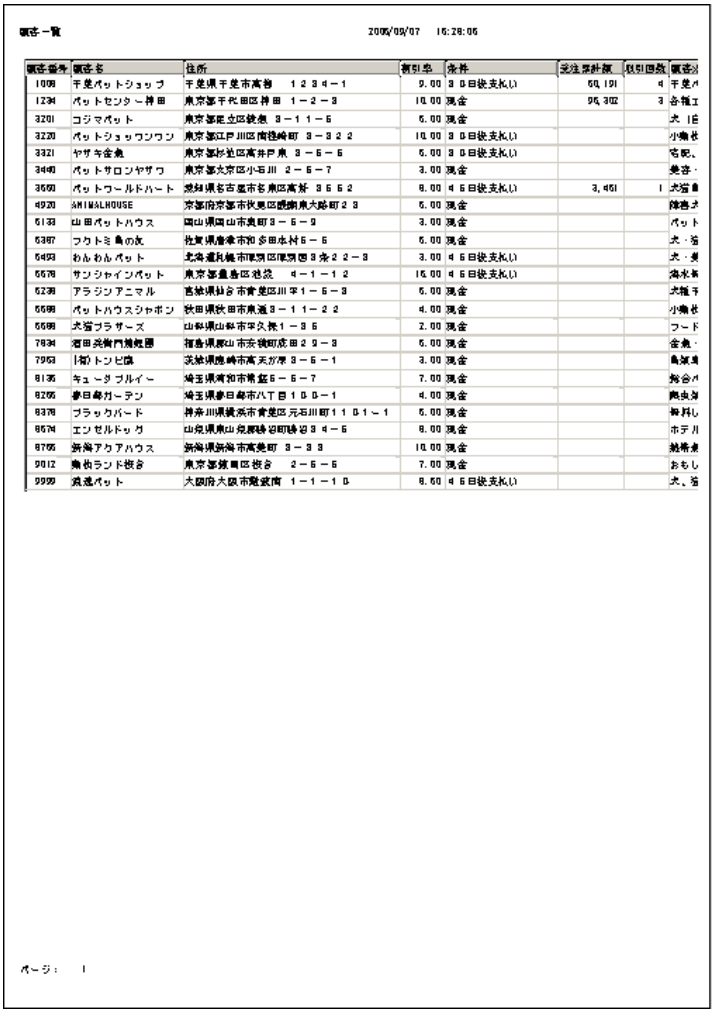


図 18-13 ページヘッダ/フッタフォームの指定（入出力特性ダイアログ）

ここで設定しておけば、ページヘッダ/フッタは自動的に印刷されるようになります。ページヘッダ/フッタのためにデータ出力コマンドを実行する必要はありません （図 18-14)。



顧客一覧　　2005/09/07　16:28:05

顧客番号	顧客名	住所	割引率	条件	受注累計額	取引回数	顧客メ
1000	千葉ペットショップ	千葉県千葉市高洲　１２３４－１	9.00	３０日後支払い	60,191	4	千葉ペ
1234	ペットセンター神田	東京都千代田区神田　１－２－３	10.00	現金	95,302	3	各種エ
3201	コジマペット	東京都足立区鹿浜　３－１１－５	5.00	現金			犬　猫
3220	ペットショップワンワン	東京都江戸川区南篠崎町　３－３２２	10.00	３０日後支払い			小動物
3321	ヤザキ金魚	東京都杉並区高井戸東　３－５－５	5.00	３０日後支払い			宅配、
3440	ペットサロンヤザワ	東京都北区小石川　２－５－７	3.00	現金			美容・
3500	ペットワールドパート	愛知県名古屋市名東区高針　３５５２	8.00	４５日後支払い	3,451	1	犬猫鳥
4920	ANIMALHOUSE	京都府京都市伏見区醍醐東大路町２３	5.00	現金			除湿ボ
5133	山田ペットハウス	岡山県岡山市奥町３－５－９	3.00	現金			ペット
5387	フクトミ亀の店	佐賀県唐津市和多田本村５－５	5.00	現金			犬・猫
5453	わんわんペット	北海道札幌市厚別区厚別西３条２２－３	3.00	４５日後支払い			犬・猫
5578	サンシャインペット	東京都豊島区池袋　４－１－１２	15.00	４５日後支払い			海水魚
5236	アラジンアニマル	宮城県仙台市青葉区川平１－５－３	5.00	現金			犬猫ネ
5598	ペットハウスシャボン	秋田県秋田市東通３－１１－２２	4.00	現金			小動物
5698	犬猫プラザーズ	山梨県山梨市平久保１－３５	2.00	現金			フード
7834	福田共働門施設園	福島県郡山市安積町成田２９－３	5.00	現金			金魚・
7953	(有)トンビ商	茨城県鹿嶋市高天が原３－５－１	3.00	現金			鳥類魚
8135	キューダブルイー	埼玉県南和市南盆５－５－７	7.00	現金			熱帯魚
8205	春日部ガーデン	埼玉県春日部市八丁目１０５－１	4.00	現金			熱帯魚
8378	ブラックバード	神奈川県横浜市青葉区元石川町１１５１－１	5.00	現金			海外し
8574	エンゼルドッグ	山梨県南山梨郡塩山和塚谷３４－５	8.00	現金			ホテル
8705	新潟アクアハウス	新潟県新潟市高美町　３－３３	10.00	現金			熱帯魚
9012	熊谷ランド板谷	東京都練馬区板谷　２－５－５	7.00	現金			おもし
9999	鈴木ペット	大阪府大阪市鶴城南　１－１－１５	8.50	４５日後支払い			犬、猫

ページ：　1

図 18-14 ページヘッダ／フッタ付きの印刷結果

18.5. テーブルコントロールの小技

図 18-14 の印刷結果では、行の幅が広すぎて、A4 の用紙に収まりきりません。このため、テーブルの幅を調整して印刷内容がすべて A4 サイズの用紙に収まるようにしましょう。また、こうすると長い項目のデータが全部表示されなくなってしまうので、長さの長い項目のカラム幅を狭めて、全体が用紙に納まるように印刷しましょう。幅を狭めたカラムについては、データの全体が見れるよう、カラムの中で折り返して複数行に分けて印刷するようにします。

18.5.1. フォーム、テーブル、行の高さ調整

今まではテーブルコントロールの 1 行には、データが 1 行づつ印刷されるようになっていましたが、長いカラムでは折り返し複数行印刷をできるようにするため、テーブルの 1 行にデータを複数行印刷できるよう、フォーム、テーブルおよび行の高さを変える必要があります。

フォーム高さの変更

フォームの高さは、フォームテーブルでフォーム特性の「高さ」で設定します。単位は cm です。

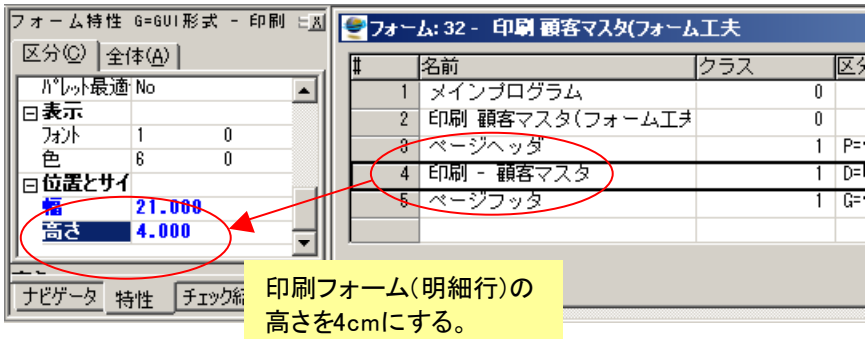


図 18-15 印刷フォーム(明細行)の高さ変更

テーブル高さの変更

テーブルの高さは、フォームエディッタ上で行います。

1. ズームしてフォームエディッタを開きます。
2. テーブルコントロールを選択します（テーブルコントロールの上をクリックすると選択されます)。
3. 下辺をマウスでドラッグして、データ部分に 4 行ほど表示できるくらいまで高を広げます。

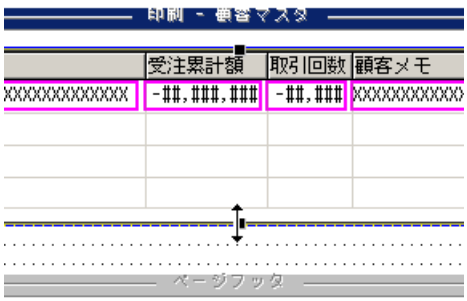


図 18-16 テーブルコントロールの高さ変更

行の高さの変更

テーブルコントロール内の、データ行の高さもフォームエディッタで行います。図 18-17 のように、テーブ

ルコントロールの行の下辺あたりにマウスカーソルを移動するとカーソルの形状が変わりますので、この状態でマウスをドラッグすると行の高さを変えることができます。1 行にデータが 4 行ほど入るように変更してください。

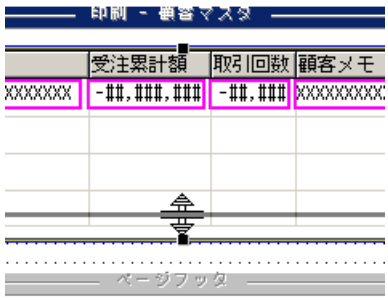


図 18-17 テーブルの各行の高さ変更

18.5.2. カラム幅の調整

次は、幅の広い項目の幅を狭め、複数行に分けてデータが印刷されるようにします。これは、

- 項目を表示するエディットコントロールのサイズ(幅と高さ)の変更
- カラムの幅の変更

という二つの手順で行います。

例として、住所のカラムをとって説明します（図 18-18)。

エディットコントロールのサイズの変更

1. フォームエディッタで、住所のデータを表示するエディットコントロールを選択します。(エディットコントロールの上をマウスでクリックすると選択できます)。
2. 右下端をマウスでドラッグして、幅と高さを調整し、横に 15 文字、縦に 4 文字くらいはいるようにします。
3. コントロール特性の「複数行編集」を Yes にします。デフォルトは No ですが、Yes にするとこのエディットコントロールのデータが長いときに行で折り返して複数行に渡って表示されるようになります。

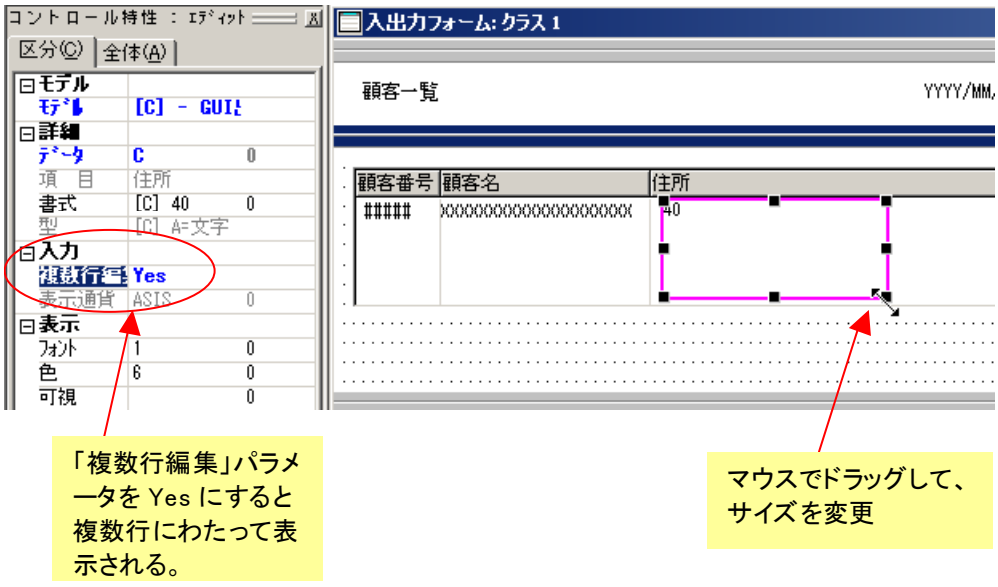


図 18-18 「住所」項目の幅変更

カラムの幅の変更

エディットコントロールのサイズを変更しただけだと、テーブルのカラムの幅は自動的に変更されません。以下のようにして、手作業で変更してやる必要があります（図 18-19 参照）。

1. 住所カラムの右側の区切り線のあたりにマウスカーソルを持っていくとマウスの形状が変わります。
2. Ctrl キーを押しながら、区切り線をドラッグし、住所のカラムの幅を狭めます。

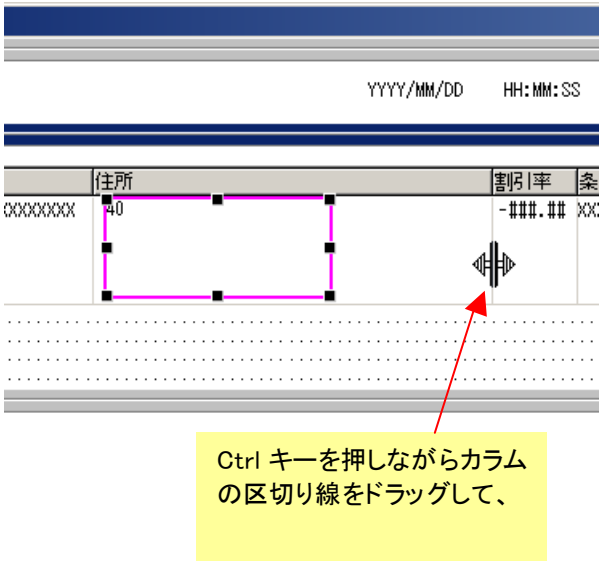


図 18-19 「住所」カラムの幅の変更

ここで、Ctrl キーを押しながらドラッグすることが重要です。Ctrl キーを押さずにそのままドラッグすると、区切り線の位置は変わりますが、割引率などそれ以降のエディットコントロールの水平位置がつられて変わって行かないので、カラムとエディットコントロールがずれてしまいます。Ctrl キーを押しながらドラッグすると、カラム幅を変えるとそれ以降のエディットコントロールの位置もそれにつれて変更されます。

同様にして、条件欄、顧客メモ欄も変更して、だいたい 21 センチくらいの幅の中にすべて納まるように設計してください。

18.5.3. 平面スタイルのテーブル

デフォルトではテーブルのスタイルが「W=Windows 立体」になっていて、このスタイルでは、

- タイトル行に立体的な陰影がつけられる。
- タイトル行のバックグラウンドが灰色に固定される。
- 行や列の区切り線が灰色になる。

となっています。

スタイルを「2=平面」とすると、出力イメージはもっと単純にすっきりとなり、印刷出力としてはこちらの方が望ましいこともあると思います。

スタイルを変更するには、テーブルコントロールのスタイル特性を変更します（図 18-20 参照）。

1. フォームエディッタで、テーブルコントロールを選択する。

   参考： 単純にテーブルコントロールの上でマウスクリックするとテーブルとともに、テーブルに属するエディットコントロールもすべて一緒に選択されてしまい、テーブルのスタイル属性を設定することができません。テーブルコントロールのみを選択するには、Ctrl キーを押しながらテーブルのタイトル行あたりをマウスクリックします。

2. 特性シートには「コントロール特性：テーブル」と出るので、「スタイル」パラメータを「2=平面」とします。

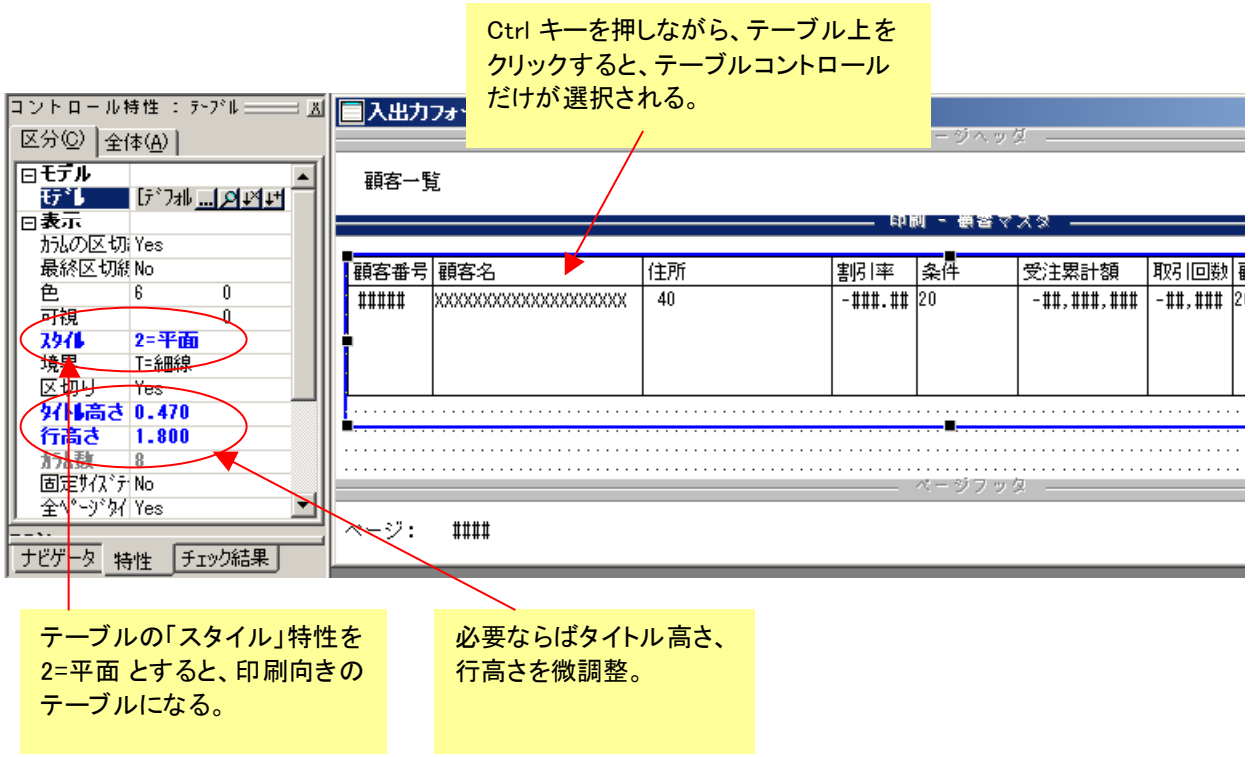


図 18-20 テーブルを平面スタイルで印刷

参考： テーブルの高さ、データ行の高さは、上述のようにマウスでドラッグすることにより調整することもできますが、テーブルコントロールを選択して、特性シートの「高さ」「行高さ」により数値で直接指定することもできます。マウスで指定しきれない微妙な調整が必要な場合には、この方法を使ってください。

18.5.4. 実行してみる

以上の変更を加えて実行してみてください。図 18-21 のような結果が得られると思います。フォームの設計とレコード数にもよりますが、1 ページには収まりきらず、2 ページ以上にまたがるかもしれません。その場合には、ページヘッダやフッタ、テーブルのタイトル行などが、2 ページ目以降にも 1 ページ目と同様に印刷されていることを確認してください。

顧客一覧　　2005/09/07　16:00:45

顧客番号	顧客名	住所	割引率	条件	受注累計額	取引回数	顧客メモ
1008	千葉ペットショップ	千葉県千葉市高柳１２３４－１	9.00	３０日後支払い	50,191	4	千葉ペットショップは１２年来のお得意様です。対応には十分に気を付けて下さい。
1234	ペットセンター神田	東京都千代田区神田　１－２－３	10.00	現金	96,302	3	各種エサ、飼育用品など取扱い
3201	コジマペット	東京都足立区綾瀬３－１１－５	5.00	現金			犬（自家繁殖あり）・美容・ホテル
3220	ペットショップワンワン	東京都江戸川区南篠崎町 ３－３２２	10.00	３０日後支払い			小動物・小鳥・観賞魚
3321	ヤザキ金魚	東京都杉並区高井戸東 ３－５－６	5.00	３０日後支払い			宅配、通信販売
3440	ペットサロンヤザワ	東京都文京区小石川 ２－５－７	3.00	現金			美容・ホテルあり
3550	ペットワールドハート	愛知県名古屋市名東区高針 ３６５２	8.00	４５日後支払い	3,451	1	犬猫鳥からエキゾチックアニマルなど何でも
4920	ANIMAL HOUSE	京都府京都市伏見区醍醐東大路町２３	5.00	現金			障害犬・猫専門
5133	山田ペットハウス	岡山県岡山市表町３－６－９	3.00	現金			ペット用品の通信販売もあり
5387	フクトミ鳥の友	佐賀県唐津市和多田本村５－５	5.00	現金			犬・猫・美容・ホテル・小動物・小鳥
5493	わんわんペット	北海道札幌市厚別区厚別西３条２２－３	3.00	４５日後支払い			犬・美容・ホテル・小動物
5678	サンシャインペット	東京都豊島区池袋４－１－１２	15.00	４５日後支払い			海水無脊椎・カメ・グリーンイグアナなども取り扱い
6238	アラジンアニマル	宮城県仙台市青葉区川平１－５－３	5.00	現金			犬種予約注文販売
6588	ペットハウスシャボン	秋田県秋田市東通３－１１－２２	4.00	現金			小動物・爬虫類・昆虫も取り扱い

ページ：　1

図 18-21 出力結果

18.6. フォントの選択

今までは、すべて同じサイズ・書体のフォントを使って印刷を行ってきましたが、グラフィック印刷の場合にはフォントを自由に変えることができます。ここでは、異なるフォントを利用する場合の方法を説明します。

18.6.1. フォントテーブル

Magic の場合、フォントはフォームから直接 Windows のフォントダイアログを使って定義するのではなく、Magic の**フォントテーブル**にいったん定義してから、フォームでその中のひとつを参照して指定する、という形になります。

フォントテーブルは、アプリケーションで使うフォントを一覧にしたもので、デフォルトでよく使うものが 110 個ほど定義されています。

フォントテーブルは、アプリケーションを閉じた状態で、メニューから 設定(S) → フォント(F) で開くこともできるし、ツールバーのフォントのアイコンをクリックして開くこともできます。(図 18-22)

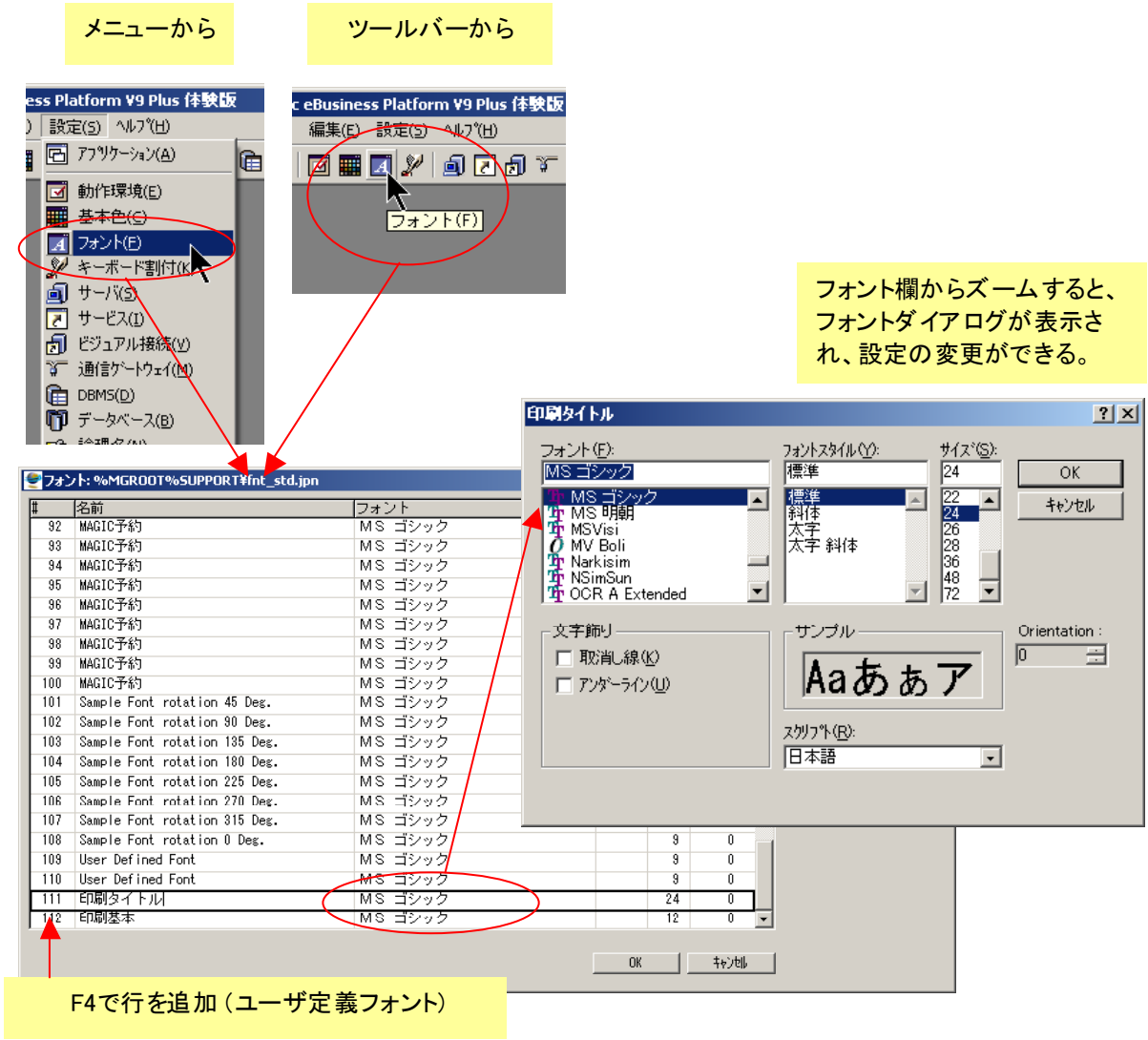


図 18-22 フォントテーブルでフォントを定義

アプリケーションで独自に定義したいフォントは、フォントテーブルの最後に追加することができます。

1. フォントテーブルを開きます。
2. カーソルを最後の行につけます。
3. F4 キーを押して、1 行追加します。
4. 名前欄には、適当な名前をつけます。この名前は Magic の中でだけ使うもので、コンマ「,」以外の任意の文字を使うことができます。
5. フォント欄からズームすると、Windows のフォントダイアログが開きますので、書体、スタイル、サイズ、飾り、角度(Orientation)などを定義してください。なお、「スクリプト」欄は「日本語」にしてください。
6. 同様にして、必要なだけのフォントを定義してください。
7. フォントテーブルで OK ボタンを押すと、**図 18-23** のような確認ダイアログが開きます。このとき、「即時有効」を Yes にしてください。デフォルトは No のままですが、No で保存すると、設定の変更がすぐに反映されず、Magic を再起動することが必要になります。

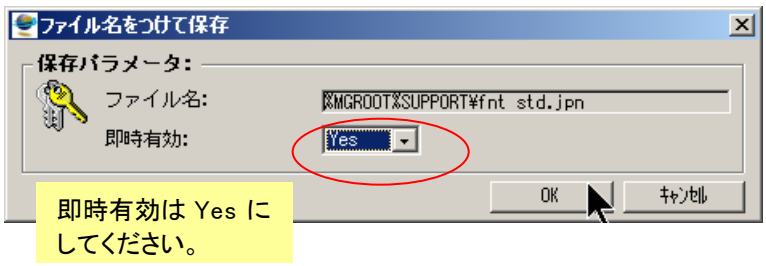


図 18-23 フォントテーブルの保存

以上でフォントの定義ができました。

18.6.2. フォント特性

フォントテーブルで定義したフォントは、フォームエディッタから「フォント」特性により番号で指定します。図 18-24 は、印刷フォームのページヘッダで、表題となるテキストコントロールで、先ほど定義したフォント 111 番を指定した例です。

フォント特性では、番号を直接キー入力することもできますし、ズームするとフォント一覧が表示されるので、ここから選択することもできます。

フォントを変更すると、フォームエディッタ上でその変更がすぐに反映されて表示されるので、結果を確認しながらフォームを設計することができます。

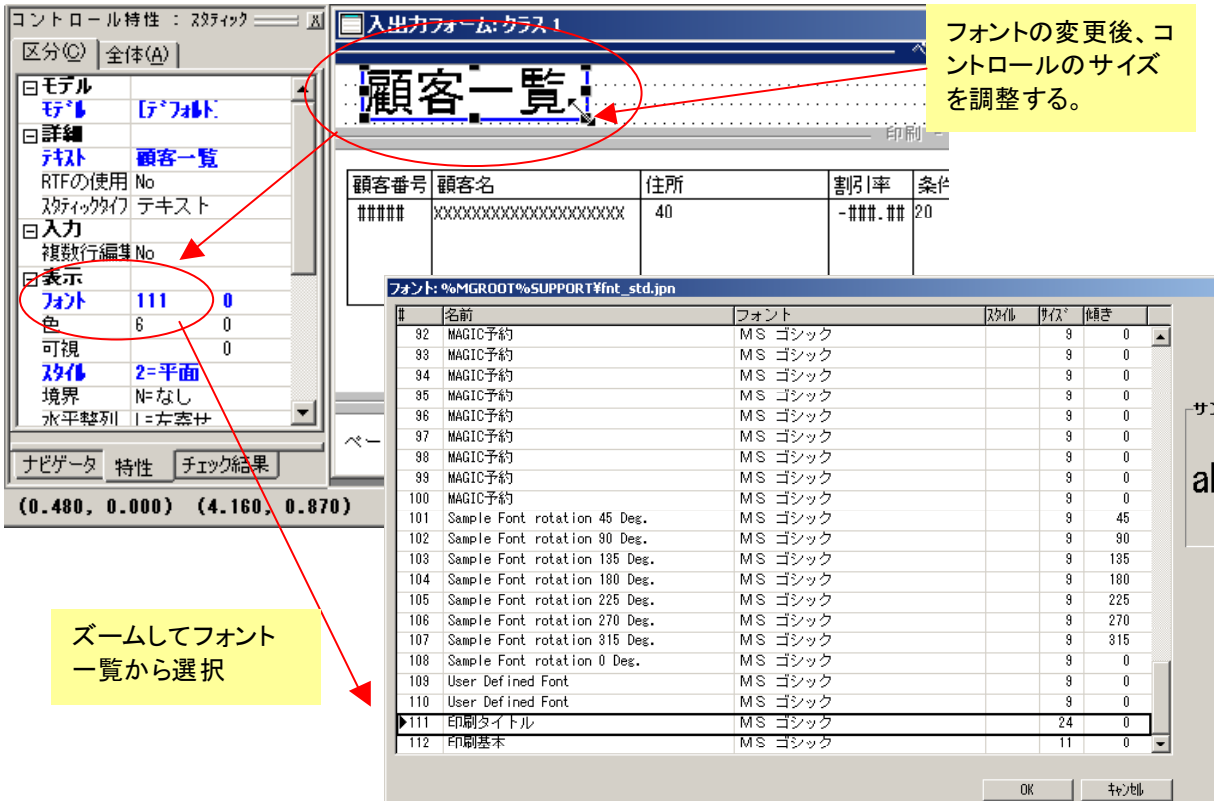


図 18-24 フォームエディッタでのフォントの指定

18.7. 受注票の印刷

最後に、図 18-25 に示すように、1 枚の受注票を 1 ページごとに印刷するプログラムを作ってみましょう。

受注一覧

2005/09/07 17

受注番号 101　　顧客番号 1008 千葉ペットショップ

受注日 2005/04/19　　千葉県千葉市高柳 １２３４－１

受注明細番号	商品番号	商品名	商品タイプ	数量	単価	合計
1	1002	ﾌﾟｰﾄﾞﾙ	D	1	10,710	10,710
2	1123	ｾﾝﾄﾊﾞｰﾅｰﾄﾞ	D	1	21,420	21,420

千葉ペットショップは１２年来のお得意様です。対応には十分に気を付けて下さい。

明細合計額	32,130
受注割引額	2,892
消費税額	877
受注合計額	30,115

図 18-25 受注票の印刷

18.7.1. 親子のタスク構造

受注票の印刷の場合には、受注入力の場合と同様、親子タスクの形にして、親タスクでは受注データをメインテーブルとし、子タスクでは受注明細データをメインテーブルとします。

プリンタ出力のための入出力ファイルは親タスクで定義し、ページヘッダ、ページフッタも親タスクで定義しておきます（図 18-26）。

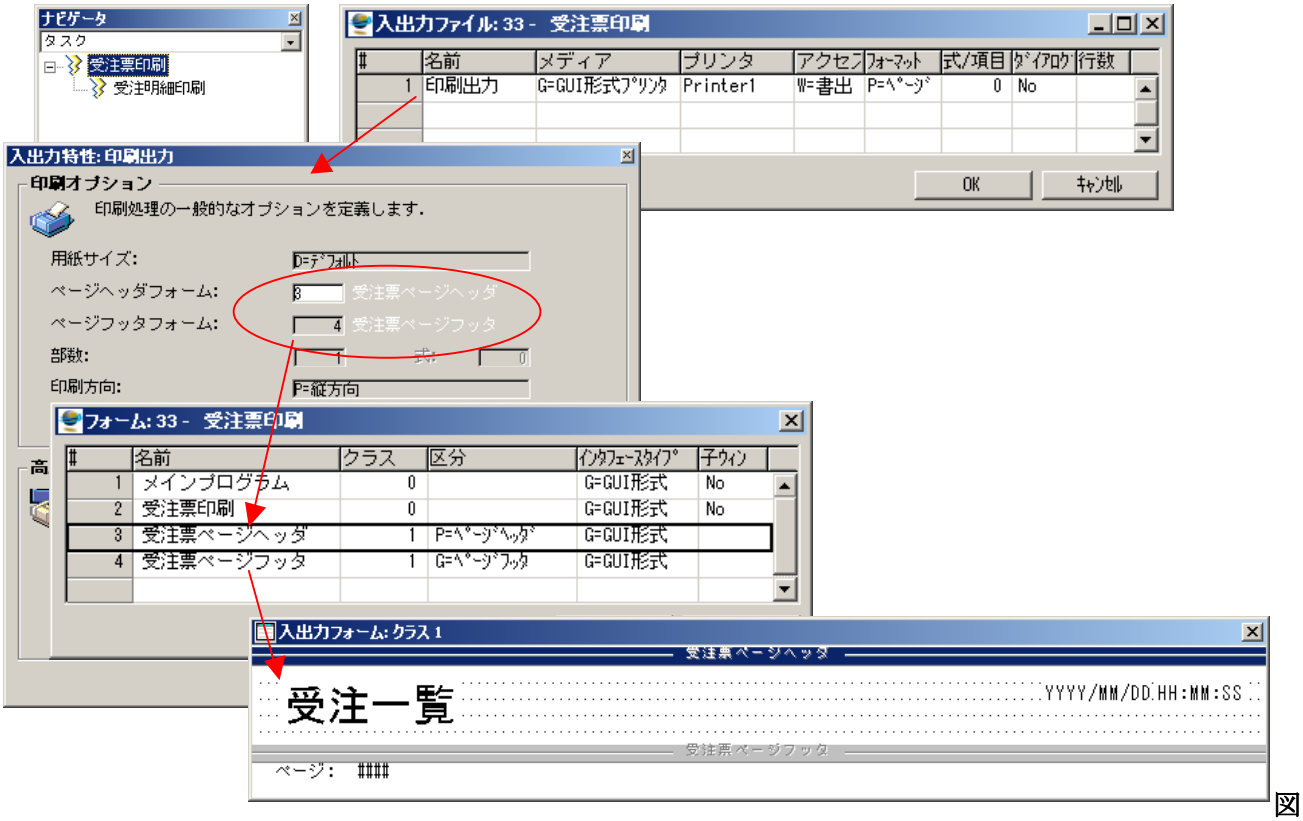


図 18-26 親タスクのフォーム

子タスクでは受注票を印刷しますが、受注票はヘッダ部分(受注番号、受注日、顧客番号、顧客名、住所を印

刷)、明細(明細行を印刷するテーブル形式)、フッタ(合計額や顧客メモなどを印刷)の 3 つの部分に分けます。親タスクのページヘッダ、ページフッタを合わせて、全部で 5 つのフォームに分けます。(図 18-27)

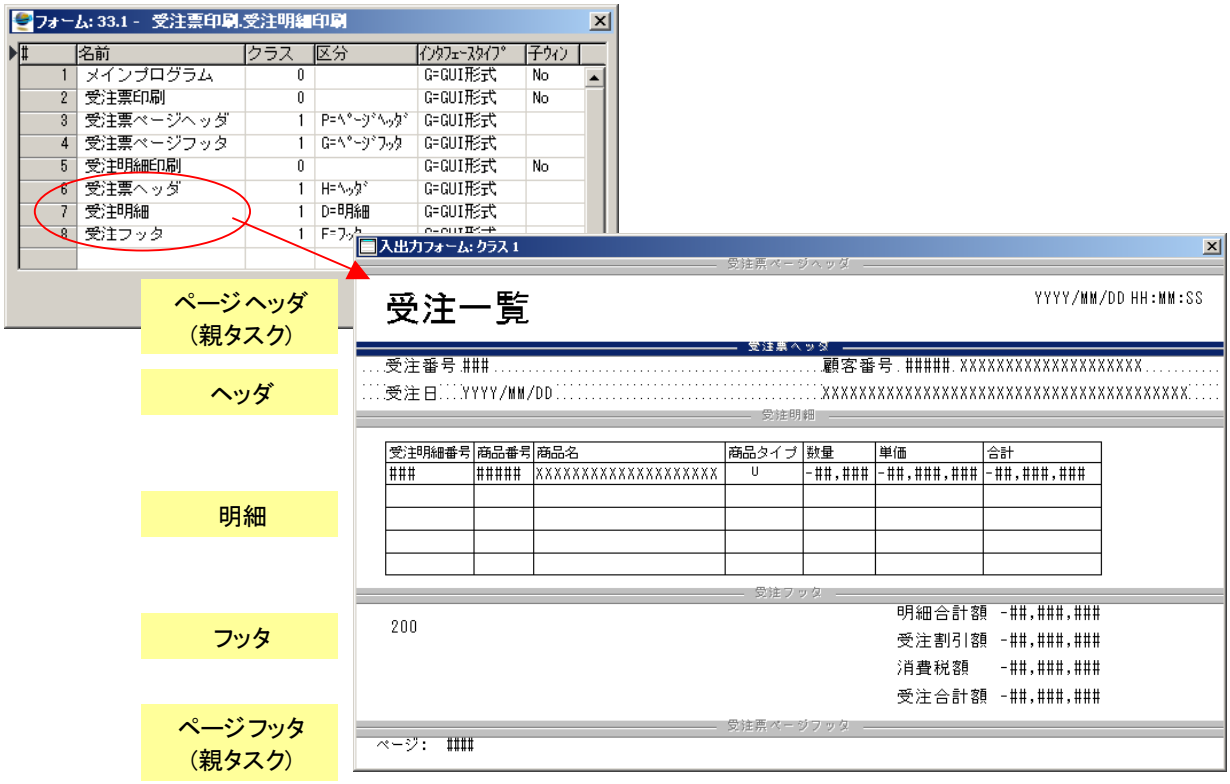


図 18-27 子タスクのフォーム

子タスクのフォームは、全体の構成に従い、次のようなタイミングで、データ出力コマンドにより印刷します。

- ヘッダ： タスク前処理で印刷。
- 明細： 各行をレコード後処理で印刷。
- フッタ： タスク後処理で印刷。

18.7.2. 固定行数のテーブル

受注票の明細行はテーブル形式で表示しますが、普通に印刷すると、この部分は明細レコード数の大小により高さが変わってしまいます。しかし印刷結果としては、レコード数に関わらず、テーブルは固定サイズで印刷したいことがあります。例えば、上の例では、明細行が 5 行に固定されています。

テーブルコントロール中のレコードサイズを固定にしたい場合には、テーブルコントロールの**固定サイズ**特性を Yes に設定します(**図 18-28**)。このパラメータはデフォルトで No になっていて、テーブルコントロールの高さはデータ出力コマンドの実行回数により自動的に拡張されていきますが、これを Yes にすると、フォームで定義されているテーブルの高さがそのまま印刷結果に反映されます。そのため、レコード数が足りない場合には空白行が埋められ、レコード数が多すぎる場合には自動的に改行して次のページに続けて印刷するようになります。

自動改ページされるときには、ページヘッダやページフッタが自動的に新しいページに付加されることは前に説明したとおりですが、その他に、「区分」で「H=ヘッダ」とされているフォーム(今の例ではフォーム#6 の「受注票ヘッダ」)も、ページヘッダに続いて自動的に印刷されます。そのため、自動的に改ページが起こった場合にも、ページのレイアウトが乱れずに出力されるようになっています。

参考： 区分が H=ヘッダのフォームが自動的に新しいページに印刷されるためには、予め 1 度はデータ出力コマンドで出力が実行されている必要があります。今の例では、タスク前処理で出力されています。一方、区分がページヘッダ、ページフッタのフォームは、入出力特性で設定しておくだけでよく、データ出力コマンドで出力を実行する必要はありません。

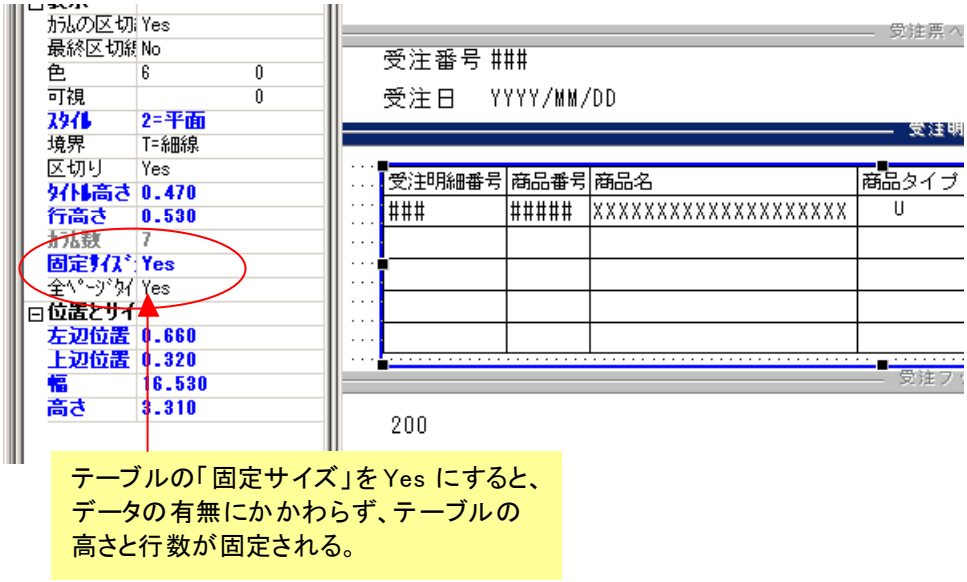


図 18-28 テーブルの行数を固定する

18.7.3. 強制改ページする

普通にデータ出力コマンドを使っている場合には、印刷出力はページが一杯になるまで同じページに出力され、一杯になったら自動的に改ページします。

しかし、受注票の印刷では、1 ページに 1 枚づつ印刷したいのですから、受注票の先頭(ヘッダ)で、ページの終わりをまたずに強制的に改ページしなければなりません。

改ページは、データ出力コマンドの「頁」パラメータで制御します。(図 18-29)

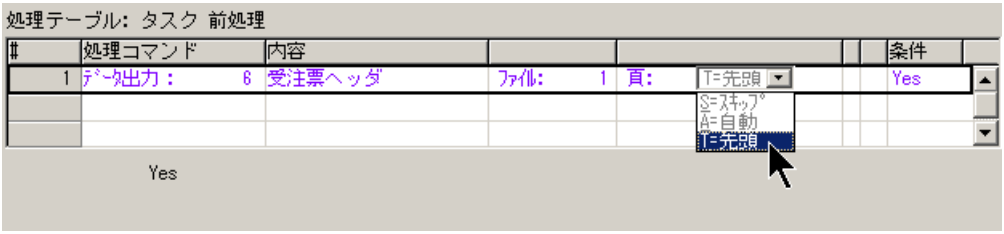


図 18-29 データ出力コマンドの「頁」パラメータ

このパラメータには表 18-1 のような三つの値があります。

「頁」パラメータの値	意味
A=自動	(デフォルト) 印刷出力はページ一杯になるまで続け、一杯になったら自動的に改ページする。
S=スキップ	ページ一杯になるまで続け、一杯になっても改ページせず、はみ出た部分は捨てる。
T=先頭	強制的に改ページし、新しいページの先頭から印刷する。

表 18-1 データ出力コマンドの「頁」パラメータの意味

強制的に改ページしてから新しいページの先頭から印刷させるために、タスク前処理でのデータ出力コマンドの「頁」パラメータは「T=先頭」に設定しておきます。

18.7.4. では作ってみましょう・・・

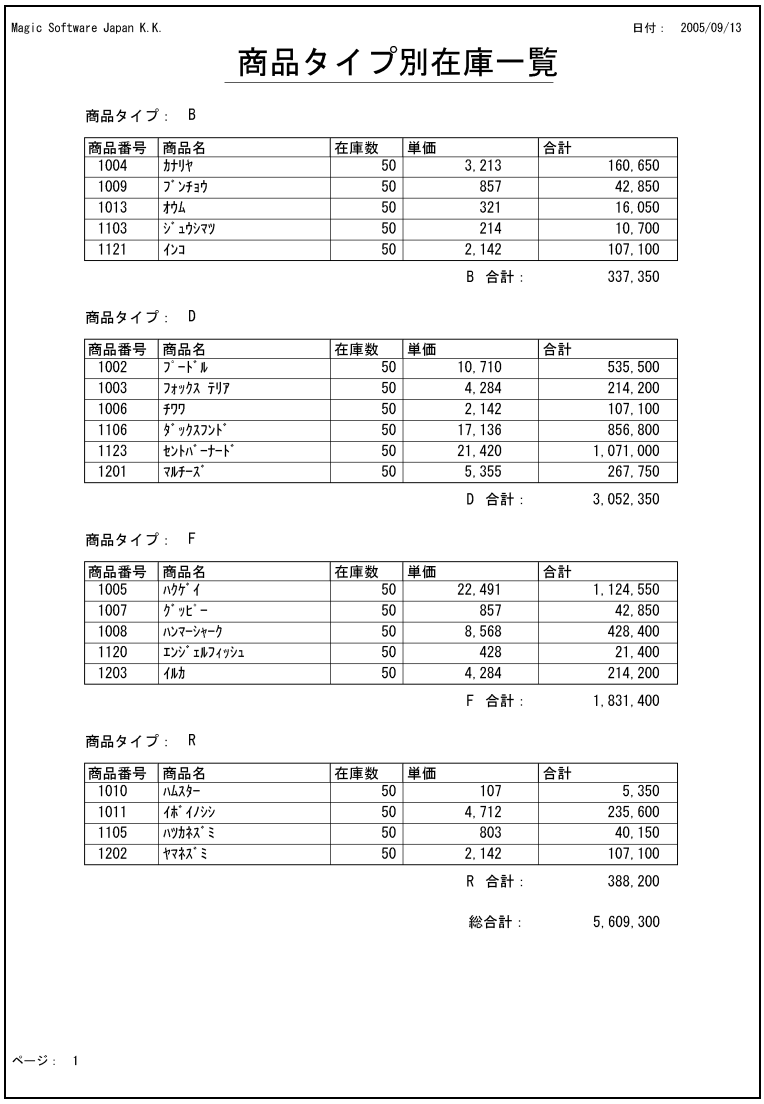
今まで学んできたことと、以上で補足したことを応用すれば、受注票の印刷は簡単にできると思います。練習問題として、ぜひ自分で作ってみてください。

参考： ここには書ききれない印刷機能の詳細に関しては、リファレンスマニュアルの次の項目も参照してください。

- 第 2 章 設定
  - 設定／プリンタ
  - 設定／フォント
- 第 6 章 プログラム
  - 入出力ファイルテーブル
- 第 7 章 処理コマンド
  - データ出力
- 第 10 章 出力フォーム
  - 入出力フォーム

19. バッチ集計

本章では、商品タイプ別在庫一覧を印刷するプログラム（図 19-1 参照）を通して、バッチ集計の方法について学んでいきます。



Magic Software Japan K.K.　　日付： 2005/09/13

商品タイプ別在庫一覧

商品タイプ： B

商品番号	商品名	在庫数	単価	合計
1004	ｶﾅﾘﾔ	50	3,213	160,650
1009	ﾌﾞﾝﾁｮｳ	50	857	42,850
1013	ｵｳﾑ	50	321	16,050
1103	ｼﾞｭｳｼﾏﾂ	50	214	10,700
1121	ｲﾝｺ	50	2,142	107,100

B 合計： 337,350

商品タイプ： D

商品番号	商品名	在庫数	単価	合計
1002	ﾌﾟｰﾄﾞﾙ	50	10,710	535,500
1003	ﾌｫｯｸｽ ﾃﾘｱ	50	4,284	214,200
1006	ﾁﾜﾜ	50	2,142	107,100
1106	ﾀﾞｯｸｽﾌﾝﾄﾞ	50	17,136	856,800
1123	ｾﾝﾄﾊﾞｰﾅｰﾄﾞ	50	21,420	1,071,000
1201	ﾏﾙﾁｰｽﾞ	50	5,355	267,750

D 合計： 3,052,350

商品タイプ： F

商品番号	商品名	在庫数	単価	合計
1005	ﾊｸｹﾞｲ	50	22,491	1,124,550
1007	ｸﾞｯﾋﾟｰ	50	857	42,850
1008	ﾊﾝﾏｰｼｬｰｸ	50	8,568	428,400
1120	ｴﾝｼﾞｪﾙﾌｨｯｼｭ	50	428	21,400
1203	ｲﾙｶ	50	4,284	214,200

F 合計： 1,831,400

商品タイプ： R

商品番号	商品名	在庫数	単価	合計
1010	ﾊﾑｽﾀｰ	50	107	5,350
1011	ｲﾎﾞｲﾉｼｼ	50	4,712	235,600
1105	ﾊﾂｶﾈｽﾞﾐ	50	803	40,150
1202	ﾔﾏﾈｽﾞﾐ	50	2,142	107,100

R 合計： 388,200

総合計： 5,609,300

ページ： 1

図 19-1 商品タイプ別在庫一覧の印刷結果

図 19-1 では、各商品の単価、在庫数に加えて、各商品タイプごとの合計も計算されて表示されています。例えば、商品タイプ B(鳥類)の合計は、「B 合計：」の横に書かれている 337,350 円です。
このような集計処理は、Magic の**グループレベル**を用いて、簡単に実現することができます。

参考情報： 本章の内容については、リファレンスマニュアルに詳しい情報がありますので参照してください。

グループレベル全般	第 5 章 Magic アプリケーションエンジン
グループレベルの定義	第 6 章 プログラム → タスク定義ウィンドウ
ソートテーブル	第 6 章 プログラム → ソートテーブル

19.1. グループレベルとは

グループレベルというのは、Magic のバッチタスクで定義される実行レベルの一つで、ある項目の値によりレコードをグループ化して集計などを行わせるために利用します。

19.1.1. グループレベルの定義

グループレベルは、タスク定義テーブルで作成します。（図 19-2）

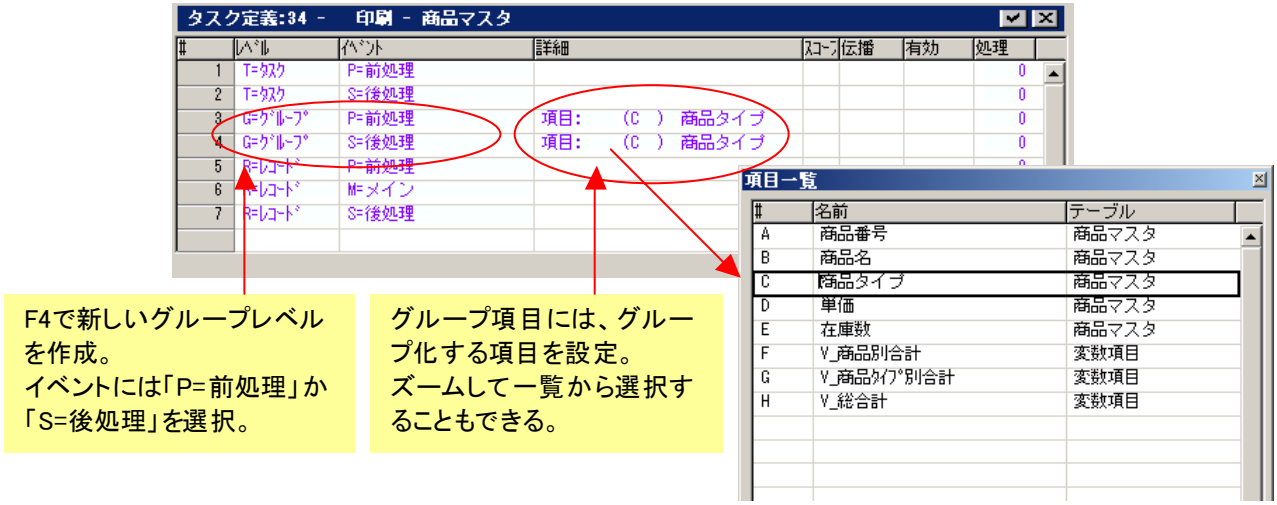


図 19-2 グループレベル

試しに、印刷プログラムを APG で作成し、グループレベルを作っていましょう。

1. プログラムリポジトリの最下行に行を追加し、APG (印刷オプション)で、商品マスタの印刷プログラムを作成してください。
2. 作成したプログラムを開き、カーソルを「T=タスク」「S=後処理」の行に置きます。
3. F4 キーを押して、新しい行を作成します。デフォルトで「レベル」は「G=グループ」で、「イベント」は「P=前処理」となっています。「イベント」は、「P=前処理」か「S=後処理」のいずれかを設定できますが、ここでは P=前処理のままにしておきます。
4. 「項目：」のカラムにカーソルを移動します。
5. F5 でズームすると項目一覧が表示されますので、「C 商品タイプ」を選択します。
6. 再度 F4 キーを押して、もうひとつ新しい行を作成します。自動的に上の行を補完する形で「レベル」は「G=グループ」、「レベル」は「S=後処理」、項目には C が設定されます。

今のところはとりあえずこれだけにしておきます。グループレベルの処理内容は後で追加してゆきます。

19.1.2. グループレベルの動作

上に見たように、グループレベルには**前処理**と**後処理**があり、**グループ項目**を設定する必要があります。
グループレベルは、次のようなタイミングと条件で実行されます。

- 最初のレコードレベルの実行前に、グループレベルの前処理が実行されます。
- あるレコードから次のレコードに移るとき、グループ項目に指定された項目の値を比較します。もし値が異なっていたら、前のレコードの値を使ってグループの後処理が実行されます。それに続き、新しいレコ

ードの値を使ってグループの前処理が実行されます。その後、通常通りにレコードレベル（レコード前処理、レコード後処理)が実行されます。これを最後のレコードまで行います。

- 最後のレコードについてのレコードレベルが終了したら、最後のレコードの値を使ってグループ後処理が実行されます。

例として、以下に商品マスタ（図 19-3）を使って、グループレベルがどのように処理されるかを説明します。

19.1.3. レコードの処理順序指定

まず、グループレベルの実行の可否は、上記に示したように、直前のレコードと現在のレコードのグループ項目の値を比較するだけで判断しています。従って、グループごとの集計などの処理を正しく行わせるためには、処理するレコードの順序を、グループ項目(今の例では商品タイプ)によってソートしておかなければなりません。図 19-3 はそのようになっています。

商品番号	商品名	商品タイプ	単価	在庫数	受注数	発注数
1004	ｶﾅﾘﾔ	B	3,213	50	3	
1009	ﾌﾝﾁｮｳ	B	857	50	1	
1013	ｵｳﾑ	B	321	50	2	
1103	ｼﾞｭｳｼﾏﾂ	B	214	50		
1121	ｲﾝｺ	B	2,142	50	1	
1002	ﾌﾟｰﾄﾞﾙ	D	10,710	50	6	
1003	ﾌｫｯｸｽ ﾃﾘｱ	D	4,284	50		
1006	ﾁﾜﾜ	D	2,142	50		
1106	ﾀﾞｯｸｽﾌﾝﾄﾞ	D	17,136	50		
1123	ｾﾝﾄﾊﾞｰﾅｰﾄﾞ	D	21,420	50	1	
1201	ﾏﾙﾁｰｽﾞ	D	5,355	50		
1005	ﾊｳｸﾞｲ	F	22,491	50	2	
1007	ｸﾞｯﾋﾟｰ	F	857	50		
1008	ﾊﾝﾏｰｼｬｰｸ	F	8,568	50	2	
1120	ｴﾝｼﾞｪﾙﾌｨｯｼｭ	F	428	50		
1203	ｲﾙｶ	F	4,284	50		
1010	ﾊﾑｽﾀｰ	R	107	50		
1011	ｲﾎﾞｲﾉｼｼ	R	4,712	50		
1105	ﾊﾂｶﾈｽﾞﾐ	R	803	50		

図 19-3 商品マスタ（商品タイプ順で並べ替え）

レコードの処理順序を指定するには、インデックスの指定と、ソートの指定の二つの方法があります。

インデックスによる指定： 商品タイプがインデックスとして定義されている場合にはそのインデックスをタスク特性で指定します。今の例では、商品マスタの第 3 番目のインデックスが、商品タイプと商品番号とを組み合わせてインデックスとして定義されているので、これを使います（図 19-4)。

注意： グループ項目は、インデックスの第一セグメントとして定義されていることが必要です。第二セグメント以降に入っていたとしても、レコード順序が正しく並ばないので利用できません。

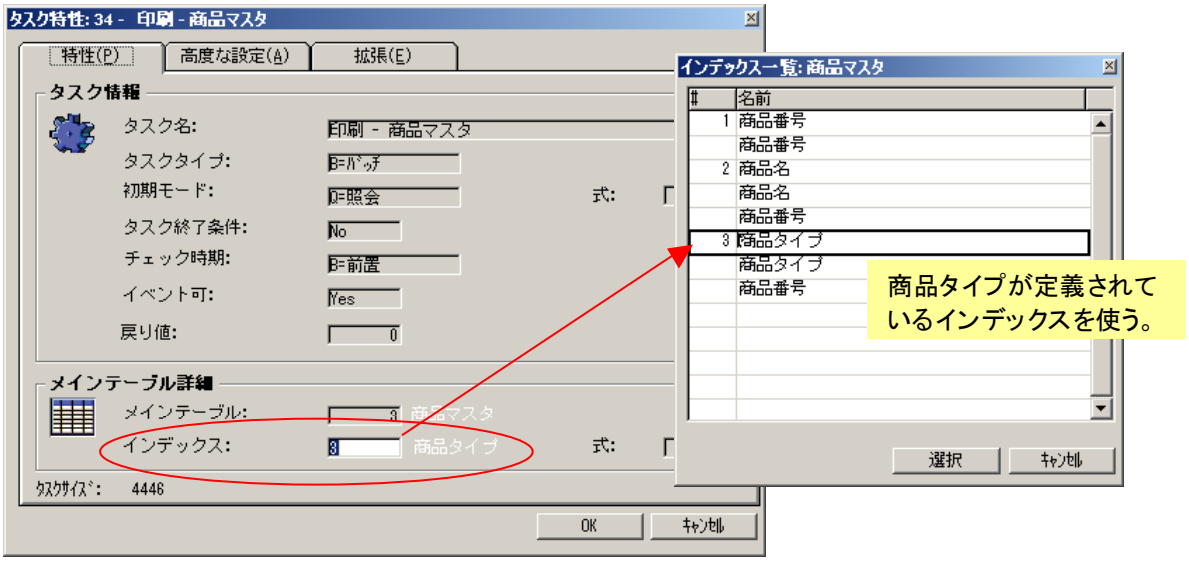


図 19-4 インデックス指定により、グループ項目順にレコード処理をさせる

ソートによる指定： もし、グループ項目が第一セグメントとして定義されているインデックスがない場合には、タスクのソートテーブルを使います。

ソートテーブルは、メニュー タスク環境(K) → ソートテーブル(S) を選択するか、あるいは Ctrl+S で開きます。ここには、ソートの順番を決める項目を設定します（図 19-5）。

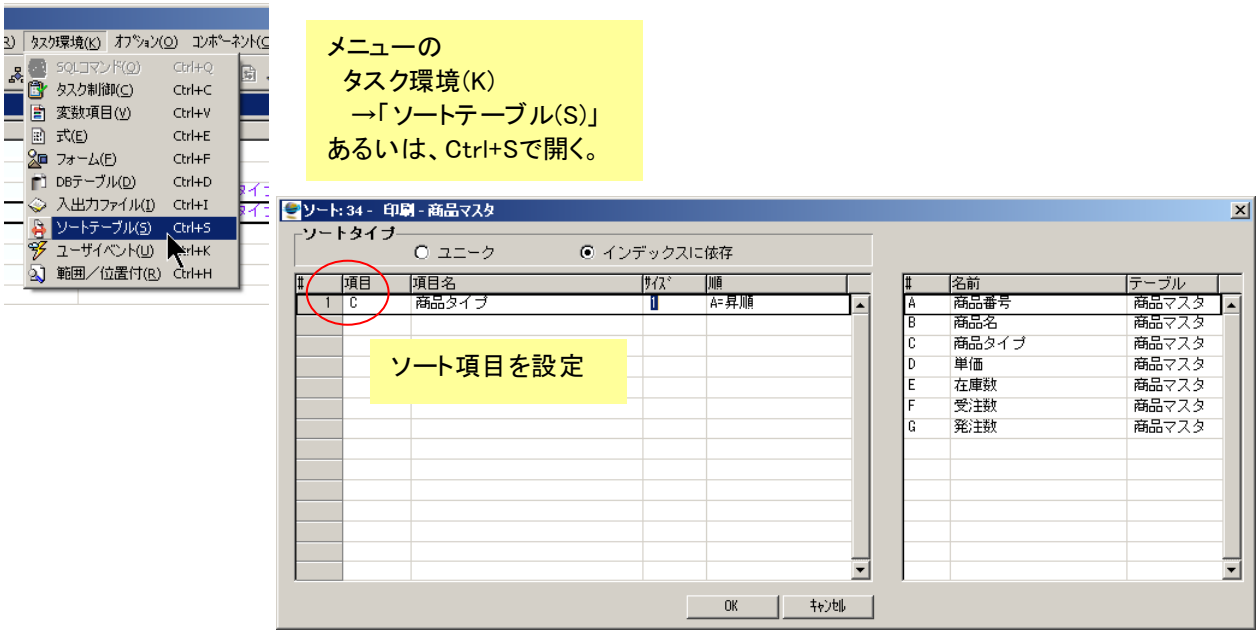


図 19-5 ソートテーブル

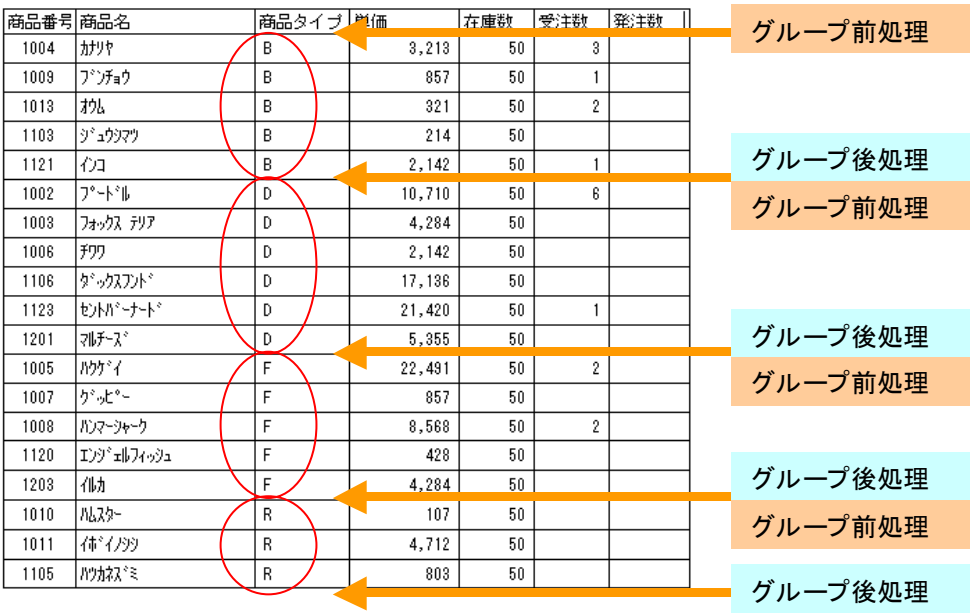
実行時、タスクにソートテーブルが指定されていると、タスクの一番最初にレコードのソートが行われます。ソートはタスクを実行するたびに毎回行われますので、レコード数によっては非常に時間がかかることがあります。

本章では、インデックスを使って項目順序を指定します。ソートについてはここでは詳しい説明を省略します

が、リファレンスマニュアル第 6 章 プログラムの「ソートテーブル」の項を参照してください。

19.1.4. 実行例

商品タイプの順に並べられたレコードを処理していく過程では、次のようにグループレベルが処理されていきます（図 19-6）。



商品番号	商品名	商品タイプ	単価	在庫数	受注数	発注数
1004	ｶﾅﾘﾔ	B	3,213	50	3	
1009	ﾌﾞﾝﾁｮｳ	B	857	50	1	
1013	ｵｳﾑ	B	321	50	2	
1103	ｼﾞｭｳｼﾏﾂ	B	214	50		
1121	ｲﾝｺ	B	2,142	50	1	
1002	ﾌﾟｰﾄﾞﾙ	D	10,710	50	6	
1003	ﾌｫｯｸｽ ﾃﾘｱ	D	4,284	50		
1006	ﾁﾜﾜ	D	2,142	50		
1106	ﾀﾞｯｸｽﾌﾝﾄﾞ	D	17,136	50		
1123	ｾﾝﾄﾊﾞｰﾅｰﾄﾞ	D	21,420	50	1	
1201	ﾏﾙﾁｰｽﾞ	D	5,355	50		
1005	ﾊｸｹﾞｲ	F	22,491	50	2	
1007	ｸﾞｯﾋﾟｰ	F	857	50		
1008	ﾊﾝﾏｰｼｬｰｸ	F	8,568	50	2	
1120	ｴﾝｼﾞｪﾙﾌｨｯｼｭ	F	428	50		
1203	ｲﾙｶ	F	4,284	50		
1010	ﾊﾑｽﾀｰ	R	107	50		
1011	ｲﾎﾞｲﾉｼｼ	R	4,712	50		
1105	ﾊﾂｶﾈｽﾞﾐ	R	803	50		

図 19-6 商品マスタでのグループ処理

1. 最初に、タスク前処理が実行されます。
2. 最初のレコード(#1004 カナリア)を読み込んだ直後に、グループレベルの前処理が実行されます。
3. #1004 のレコードレベルの処理(レコード前処理、レコード後処理)が実行されます。
4. 次のレコード(#1009 ブンチョウ)を読み込みます。この場合、前のレコードの商品タイプと今のレコードの商品タイプは同じ B ですので、グループ後処理、前処理は実行されません。
5. #1009 のレコードレベルの処理(レコード前処理、レコード後処理)が実行されます。以下、#1121 インコまで同じです。
6. #1121 インコの後、#1002 プードルが読み込まれたとき、商品タイプの比較が行われますが、このときには前のレコードが B であるのに、今のレコードは D となるので、ここでグループの区切りが来たこと判定されるので、前のレコード(#1121 インコ)のデータを使って、グループ後処理が実行されます。つづいて、新しいレコード(#1002 プードル)のデータを使ってグループ前処理が実行されます。その後、#1002 プードルについて、レコードレベルの処理が実行されます。
7. 以下同様にして、#1201 マルチーズと#1005 ハクゲイの間、#1203 イルカと#1010 ハムスターの間で、グループの区切り(グループ項目の値が異なる)がありますから、グループ後処理と前処理が実行されます。
8. 最後のレコード#1105 ハツカネズミの後で、グループ後処理が実行されます。
9. 最後にタスク後処理が実行され、このタスクが終了します。

19.2. 集計処理を追加するには

グループレベルを利用して、集計処理を追加するには、どうすればよいでしょうか？

ここでは集計する値として、商品タイプごとの在庫額(単価×在庫数)の合計と、総在庫額とを計算します。

- まず、これらの合計額を格納する変数をレコードメインで定義しておきます。
- 総在庫額は、タスク前処理でゼロにリセットし、各レコードごとにレコード後処理で単価×在庫数を加算していきます。最後にタスク後処理で値を印刷させます。
- 商品タイプごとの在庫額については、各レコードグループごとに計算し、印刷しなければならないので、グループレベルを使って処理します。すなわち、グループ毎の合計値はグループ前処理でゼロにリセットし、各レコードごとにレコード後処理で単価×在庫数を加算し、グループ後処理で値を印刷させるようにします。

これをプログラムにすると、次のようになります。フォームはまだ説明していませんが、次の項で定義します。

```
タスク特性
  名前： 商品タイプ別在庫一覧
  タスクタイプ： B=バッチ
  初期モード： Q=照会
  メインテーブル： 3 商品マスタ
  インデックス： 3 商品タイプ

入出力ファイル
 #1  名前： プリンタ出力    メディア： G=GUI プリンタ
   ページヘッダフォーム： 3 ページヘッダ
   ページフッタフォーム： 8 ページフッタ

タスク前処理
  項目更新  V_総合計  式： 0

タスク後処理
  データ出力  7 総合計フッタ  ファイル： 1

グループ前処理 項目： C 商品タイプ
  項目更新  V_商品タイプ別合計  式： 0
  データ出力  4 商品タイプ別ヘッダ  ファイル： 1

グループ後処理 項目： C 商品タイプ
  データ出力  6 商品タイプ別フッタ  ファイル： 1

レコードメイン
  セレクト R=実データ  1 商品番号
  セレクト R=実データ  2 商品名
  セレクト R=実データ  3 商品タイプ
  セレクト R=実データ  4 単価
  セレクト R=実データ  5 在庫数
  セレクト V=変数  1  V_商品別合計  モデル： 17 合計(10 桁)   代入： 単価 * 在庫
  セレクト V=変数  2  V_商品タイプ別合計  モデル： 17 合計(10 桁)
  セレクト V=変数  3  V_総合計  モデル： 17 合計(10 桁)

レコード後処理
  項目更新  V_商品タイプ別合計  式： 商品タイプ別合計＋V_商品別合計
  項目更新  V_総合計  式： V_総合計＋V_商品別合計
```

19.3. フォームの定義

印刷フォームとしては、次のものを用意します。

- ページヘッダとフッタ　（必須ではありませんが、体裁を整えるため）
- グループレベルの集計結果を印刷するためのヘッダとフッタ
- 総合計を最後に印刷するためのフッタ
- レコードごとのデータ印刷のための明細フォーム

フォームテーブルは図 19-7 のようになります。

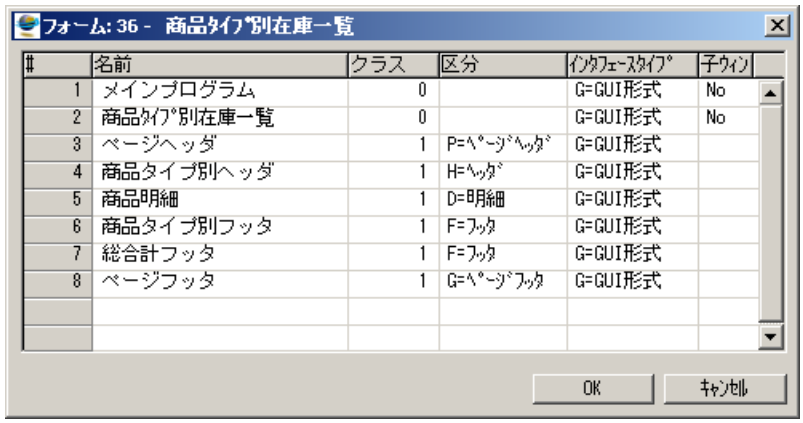


図 19-7 フォームテーブル

これらの印刷用のフォームは次のような形（図 19-8）で設計しましょう。このフォームの設計に関して特別なことはないので、詳細な説明は省略します。

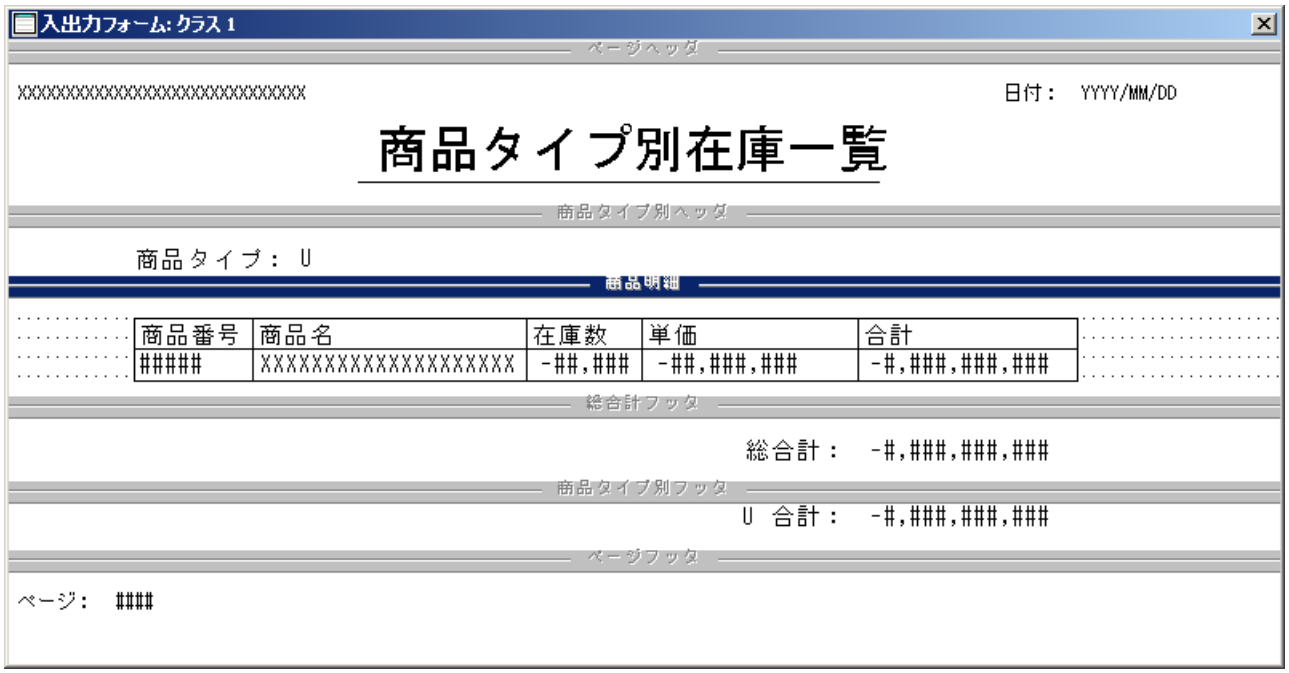


図 19-8 印刷用フォームの最終形

以上でプログラムができあがりです。実際に実行してみて、正しい結果が印刷されるか確認してください。

20. リポジトリ入出力

Magic のアプリケーションの定義(モデル、テーブル、プログラムなどのリポジトリ定義)は、MCF ファイルというバイナリファイルにすべて格納されています。今作っているペットショップデモの MCF ファイルは、petshop_ctl.mcf という名前で Magic ディレクトリに作成されているものです。このファイルは、デフォルトの設定では Pervasive.SQL のファイル形式であり、データ内容は Magic のバージョンに依存した内部バイナリ形式で格納されています。

このような形式でアプリケーションの定義を格納することにより、コードレス・コンパイルレスでアプリケーションの実行を高速に行うことができる利点があるのですが、他方、次のような欠点もあります。

- なんらかの理由で MCF ファイルが万一破損してしまった場合、回復の方法がない。
- データ内容は Magic のバージョンに依存しているため、Magic をバージョンアップした場合に互換性がなく、MCF ファイルを読むことができなくなる。
- プログラムなどの作成・修正・削除を繰り返していくと、MCF のデータサイズが必要以上に大きくなってしまうことがある。

このような理由から、バイナリ形式の MCF ファイルのほかに、テキスト形式のファイルがあると保存やバージョンアップの対応などのために便利です。

Magic では、リポジトリ入出力の機能を備えており、リポジトリ出力によりバイナリの MCF ファイルをテキストファイルとして出力し、逆に、リポジトリ入力によりテキスト形式に出力したアプリケーションを MCF ファイルに読み込むことができます。

Magic コミュニティの中では、サンプルや便利なツール類などが Magic アプリケーションとして公開・共有されていますが、普通それらは、Magic のバージョンが変わっても利用できるよう、リポジトリ出力形式で公開されています。そういう意味では、リポジトリ出力形式は、Magic アプリケーションの**ソースファイル**とも言うことができます。

参考情報： 本章の内容については、リファレンスマニュアルに詳しい情報がありますので参照してください。 リポジトリ入出力全般　　第 19 章 ユーティリティ → リポジトリ入出力ユーティリティ

20.1. リポジトリ出力形式

Magic アプリケーションを MCF からテキスト形式に出力するには、**リポジトリ出力機能**を使います。出力されたテキストファイルのデータを、**リポジトリ出力形式**と呼びます。

リポジトリ出力機能は、次のようにして使います。

1. メニュー ファイル(F) → リポジトリ入出力(X) を選ぶか、あるいは Shift+F10 キーを押して、リポジトリ入出力ダイアログを開きます。ツールバーからリポジトリ入出力のアイコンをクリックすることもできます。
2. 「操作」としては、「E=出力」を選びます。
3. 「リポジトリタイプ」としては、出力したいリポジトリを選択します。「A=アプリケーション」を選ぶと、アプリケーションの全リポジトリが出力されます。
4. リポジトリタイプとして、モデル、テーブル、プログラムなどを選択した場合には、番号の範囲を指定して出力する対象を絞ることもできます。絞り込みたい場合には、「開始番号」と「終了番号」を指定します。ズームして一覧から選択することもできます。デフォルトでは、全部が対象となっています。
5. 出力するリポジトリ出力形式のテキストファイル名を「ファイル名」に指定します。
6. OK ボタンを押すと、出力が開始されます。

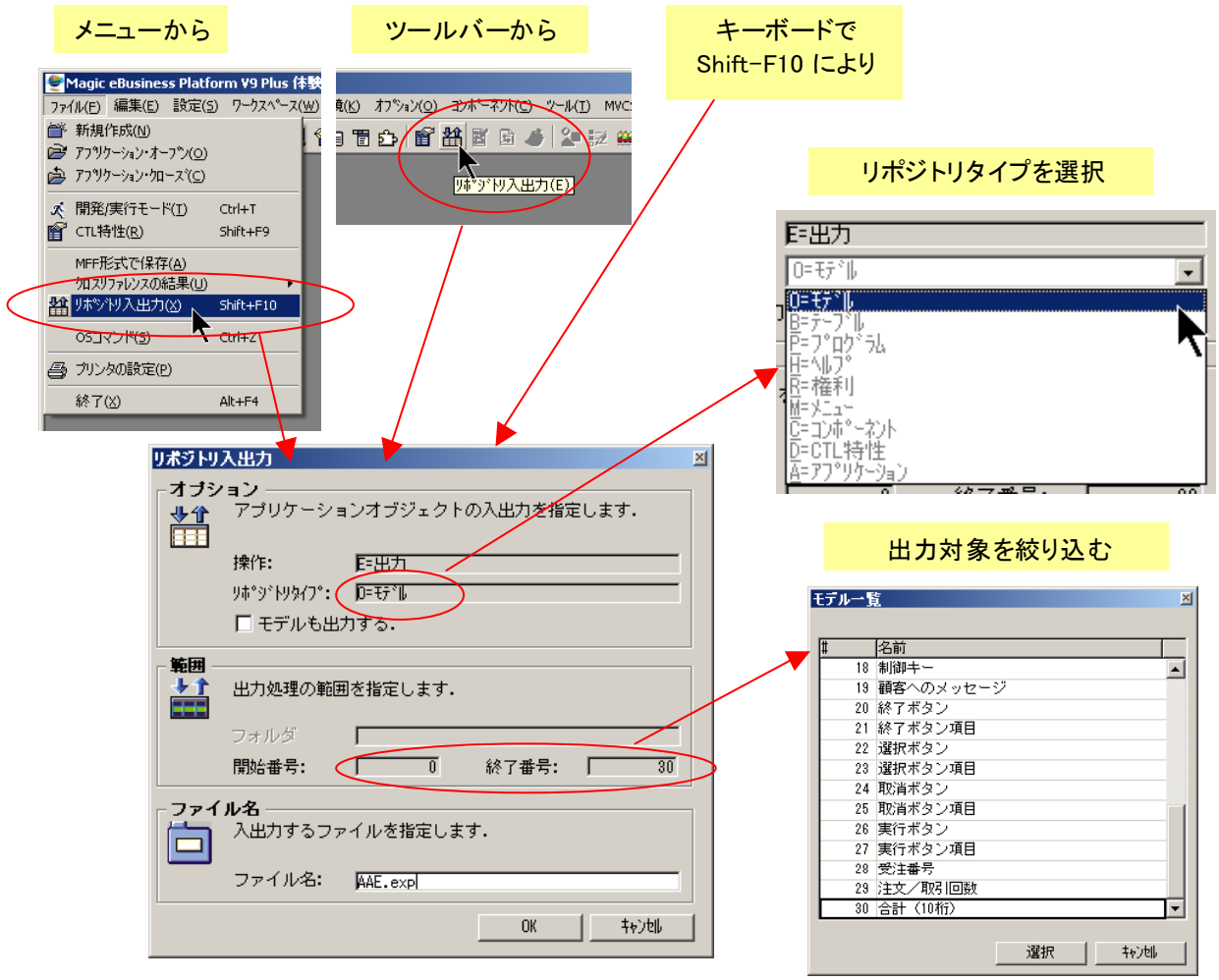


図 20-1 リポジトリ入出力ダイアログ

リポジトリ出力形式のファイルの中身は、図 20-2 のようなものです。

```
VRSN=940.03
APPLICATION=Y
MODEL=Y
TABLE=Y
PROGRAM=Y
MAIN_PROGRAM=Y
HELP=N
RIGHT=N
MENU=Y
COMPONENT=N
APP_PROP=Y
ISN=N
WITH_MODELS=N
MAIN_PRG_VARS=0
MAIN_PRG_DSPS=1

MODELS={DESC="顧客番号",
PROPLIST={MODEL="FIELD",
PROP={ID="MODEL",DATA={MODEL_ID={DEF_OBJ="FIELD_NUMERIC"}}},
PROP={ID="SELECT_PROGRAM",DATA={PUBLIC_OBJ={OBJ=10}}},
PROP={ID="PICTURE",DATA={VAL="5Z"}},
PROP={ID="NULL_VALUE"},
PROP={ID="DEC"},
PROP={ID="WHOLE",DATA={VAL=5}},
PROP={ID="NEGATIVE"}}},
・・・・
```

図 20-2 リポジトリ出力形式ファイル

このファイルはテキストファイルではありますが、人間が読んで理解・修正・作成することを目的としているわけではなく、あくまでバイナリの MCF ファイルの別形式として定義されているものですので、各キーワードや値の意味については公開されていません。

注意： 非サポートの裏技として、テキストエディッタでリポジトリ出力ファイルを修正することは可能ですが、一つの値は他の値と依存関係があることがあり、それを良く把握せずに修正すると全体の整合性を崩してしまう可能性があるため、推奨されません。修正する場合には、自己責任において行ってください。

参考： 通常のバックアップを目的とする場合には、「リポジトリタイプ」として「A=アプリケーション」を指定して、アプリケーション全体をリポジトリ出力するようにしてください。それ以外(モデル、テーブル、プログラム)に設定して、アプリケーションの一部分のみを切り出してリポジトリ出力をすることも可能ですが、リポジトリ入力時に注意して入力しないと、不完全なリポジトリ入力となり、整合性が壊れたり、最悪の場合には Magic の異常終了などが起こることがあります。

20.2. リポジトリ入力

リポジトリ入力はリポジトリ出力とは逆に、リポジトリ出力形式のファイルを入力して、アプリケーションに追加する機能です。
リポジトリ入力は、リポジトリ出力と同様、リポジトリ入出力ダイアログを用いて行います。（図 20-3）

1. メニュー ファイル(F) → リポジトリ入出力(X)、あるいはキー入力 Shift+F10 により、リポジトリ入出力ダイアログを開きます。ツールバーのリポジトリ入出力アイコンをクリックしても開きます。
2. 「操作」は「I=入力」とします。
3. 「ファイル名」には、リポジトリ出力形式のテキストファイル名を指定します。
4. OK ボタンを押すと、リポジトリ入力が始まります。

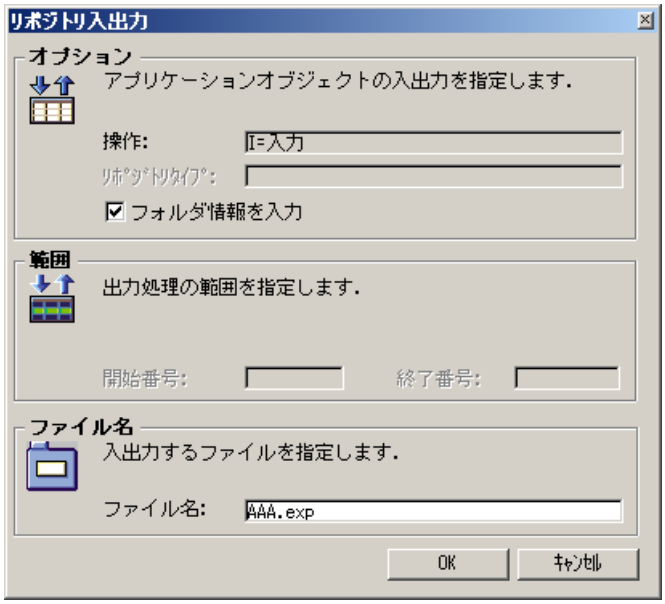


図 20-3 リポジトリ入力

リポジトリ入力された内容は、既存のリポジトリにそれぞれ追加されます。ただし、メニューリポジトリだけは、追加するか上書きするかを選択するメニューが出ます。
従って、バックアップの目的でアプリケーション全体をリポジトリ出力した場合、再度復元する場合には、空のアプリケーションを作成して、それにリポジトリ入力するようにしなければなりません。

参考： リポジトリ入力は、利用中の Magic よりも古いバージョンの Magic で出力したものでも入力することができます。このため、Magic 本体をアップグレードする場合には、必ずリポジトリ出力 → リポジトリ入力を行います。
ただし、dbMAGIC V8.2 でのリポジトリ出力形式を入力する場合には、いくつかの注意点があります。詳細についてはここでは説明しませんが、製品添付のリファレンスマニュアルや readme.chm ファイルなどのマイグレーションに関する項目を参照してください。また、dbMAGIC V7 以前のバージョンからのリポジトリ入力はサポートされていません。

21. クロスリファレンス

Magic eDeveloper には、強力なクロスリファレンス機能があります。クロスリファレンス機能を活用することにより、アプリケーションオブジェクトの依存関係をすばやく確認したり、修正したりするのが容易になります。

ここでは、クロスリファレンスのごく基本的なことのみ説明します。詳細については、リファレンスマニュアルの第 19 章 ユーティリティの章の クロスリファレンスユーティリティ を参照してください。

参考： このリファレンスマニュアル第 19 章 ユーティリティの章には、クロスリファレンスを始めとして、Magic を便利に使うための多くのユーティリティが説明してあります。クロスリファレンス以外の部分も一読されることをお勧めします。

21.1. 基本的な使い方

Magic のリポジトリで定義されているものは、ほとんどどんなものでもクロスリファレンスの対象になります。最初に例として、モデルリポジトリで定義されている「顧客番号」モデルがアプリケーションのどこで参照されているかを調べて見ましょう 。

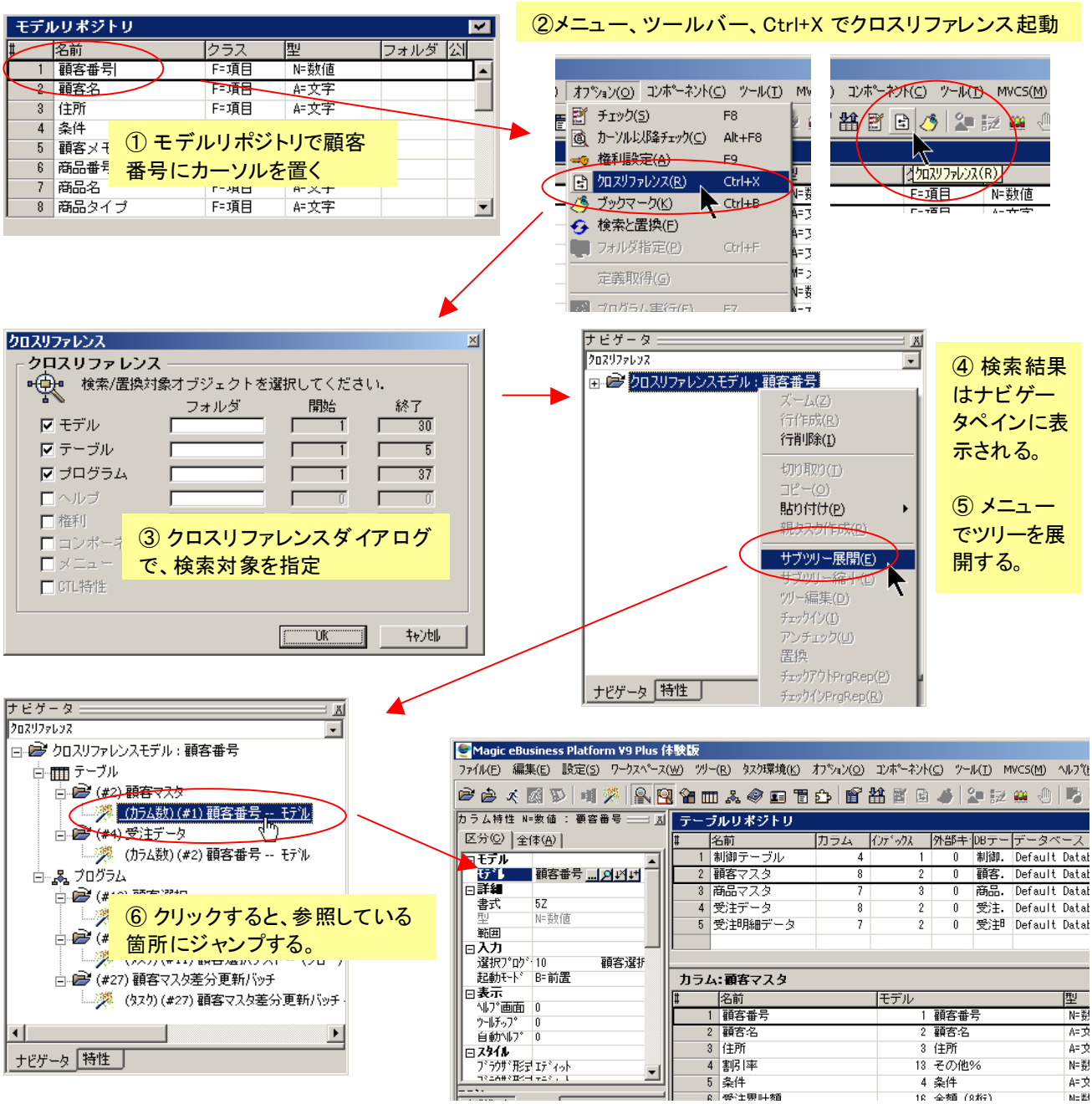


図 21-1 クロスリファレンスの使用法

① モデルリポジトリを開き、「顧客番号」の行にカーソルを置きます。

② 次のいずれかにより、クロスリファレンスを起動します。

(ア) メニュー オプション(O) → クロスリファレンス(R) を選択

(イ) ツールバーより、クロスリファレンスアイコンをクリック

(ウ) Ctrl+X キーを入力

③ クロスリファレンスダイアログが開きますので、検索対象を狭めたい場合には、リポジトリの種類、

検索範囲(範囲を開始番号と終了番号で指定)を指定することができます。デフォルトではアプリケーション全体が検索の対象となります。

④ OK ボタンを押すと、ナビゲータペインに結果がツリー形式で表示されます。

⑤ 最初はツリーが閉じた状態で表示されるので、メニューで「サブツリー展開」を選ぶと、ツリーが最後まで展開されます。ツリー上で、[+] 、[−] ノードをクリックして一つづつ展開することもできます。

⑥ 展開されたノードをクリックすると、参照箇所にジャンプします。上の例は、テーブルリポジトリの「顧客マスタ」テーブルの「顧客番号」カラムで、モデルとして参照されているところにジャンプしたところです。

21.2. クロスリファレンスできるもの

Magic V9Plus では、クロスリファレンスできるものが大幅に拡張されました。例えば、主なものには次のようなものがあります。

- モデル
- テーブル
- テーブル中の次のオブジェクト
  - カラム
  - キー
- プログラム
- プログラム中の次のオブジェクト
  - 変数
  - 式
  - ユーザイベント
  - フォーム
  - 入出力ファイル
- ヘルプ

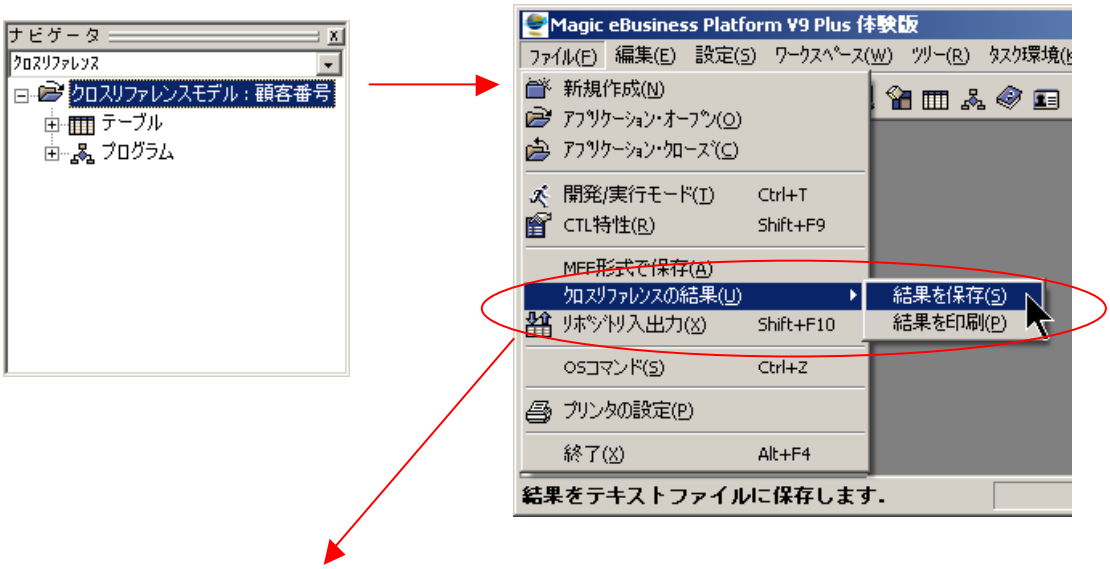
これ以外にも多くのクロスリファレンスが可能です。詳細については、リファレンスマニュアル第 19 章ユーティリティの クロスリファレンス の項を参照してください。

21.3. クロスリファレンスの保存

クロスリファレンスの結果は、アプリケーションを閉じるときに自動的に保存されます。保存先は、「マイドキュメント」ディレクトリの直下に、「.XRF」という拡張子で作成されます。このため、Magic をいったん終了し、再起動してアプリケーションをオープンしたときでも、ナビゲータペインの「クロスリファレンス」には、前回のクロスリファレンスの結果が残っています。

この .XRF ファイルは、バイナリの内部形式で格納されているので、テキストエディッタなどで見ることができません。しかし、クロスリファレンスの結果をテキストファイルとして保存しておきたいことがあると思います。そのようなときには、クロスリファレンスの保存メニューを使います(図 21-2)。

1. ナビゲータペインで、「クロスリファレンス」を表示させ、結果の一つを選択します。
2. メニュー ファイル(F) → クロスリファレンスの結果 (U) → 結果を保存(S) を選択します。
3. ファイルダイアログが開くので、ファイル名を指定します。
4. 指定されたファイルに、クロスリファレンスの結果がテキストで格納されます。




```
クロスリファレンスモデル：顧客番号
テーブル：
        (#2) 顧客マスタ
                (カラム数) (#1) 顧客番号 -- モデル
        (#4) 受注データ
                (カラム数) (#2) 顧客番号 -- モデル
プログラム：
        (#10) 顧客選択
                (タスク) (#10) 顧客選択 -- (フロー) (#4) レコード メイン -- 項目 -- (#1) P_顧客番号 -- モデル
        (#11) 顧客選択テスト
                (タスク) (#11) 顧客選択テスト -- (フロー) (#4) レコード メイン -- 項目 -- (#1) 顧客番号 -- モデル
        (#27) 顧客マスタ差分更新バッチ
                (タスク) (#27) 顧客マスタ差分更新バッチ -- (フロー) (#4) レコード メイン -- 項目 -- (#1) P_顧客番号
```

図 21-2 クロスリファレンスの結果の保存

22. おわりに

最後までチュートリアルを読んでいただきまして、ありがとうございました。Magic の高い生産性と保守性の秘密を、実習を通して感じていただけましたでしょうか？

本書で説明した機能は、Magic の持つ機能の中のごく基本でしかありません。本書で触れられなかった内容で、他に重要なものが、いくつもあります。

- オンラインタスクでのイベントドリブンプログラミング（本書ではごく基本的な事柄だけ触れました)
- 豊富な組み込み関数。
- RDBMS (Oracle, MS-SQL、DB2UDB)のサポート
- トランザクション処理のサポート
- Magic コンポーネント
- プログラム保守のための機能（データ再編成、モデルの変更の継承)
- Magic Application Server (サーバ機能)
- Web アプリケーションの作成（マージとブラウザクライアント)
- XML や Web サービスのサポート

これらについては、製品添付のマニュアル、開発者ガイドの他、弊社より発行している自習書や、開発者向けセミナーなどで詳しく紹介しておりますので、ぜひとも Magic をご愛用くださいますようお願いいたします。

Magic eDeveloper™ Plus V9