

設計・製造支援アプリケーション構築プラットフォーム

MZ Platform

購買管理アプリケーション作成
チュートリアル



独立行政法人
産業技術総合研究所

= 注意事項 =

独立行政法人 産業技術総合研究所(以下、産総研)は、本文書の記載内容に関して、一切の明示的または暗示的な責任または保証を負うものではありません。

本文書の一部または全部を、産総研の文書による承諾なく、無断で複写、複製、転載することを禁じます。本書の情報の使用に対して特許責任は一切負いません。また、本書は細心の注意を払って作成されていますが、産総研は、本書に掲載されている情報の正確性、誤り、記載漏れについての責任を負うものではありません。また、本書に掲載されている情報は、将来予告なしに変更される場合があります。

MZ Platform をいかなる第三者に対しても、産総研の許可なく貸与、譲渡、その他の方法で使用させたり、担保に供したりしてはなりません。また産総研の許可なく、別のハードウェアに複製を作成することも禁じます。

本書中の社名、商品名などは、各社の登録商標または商標です。

= 目次 =

1. 概要	1
2. データベース連携部の作成	4
2.1. 「ローカルMySQL管理」複合コンポーネントの保存	5
2.2. 「ローカルMySQL管理」複合コンポーネントのロード	7
2.3. 「ローカルMySQL管理」複合コンポーネントの設定	11
2.4. 動作確認	17
2.5. SQL文実行機能の追加	23
2.6. 複数行一括追加機能の追加	30
2.7. バックアップ機能の追加	35
2.8. データベース内容設定機能の追加	42
2.9. データベーステーブル編集機能の追加	49
3. 発注依頼部の作成	59
3.1. 購買管理データベースの作成	59
3.2. 「発注依頼」複合コンポーネントの作成	61
3.3. 初期コンポーネントの追加	62
3.4. 公開メソッドおよびコンポーネント接続の初期設定	64
3.5. トップ階層におけるイベント接続先の初期設定	68
3.6. 発注依頼フレームのメニュー登録	71
3.7. 参照機能の作成と登録機能作成の準備	74
3.8. 登録機能の作成	87
3.9. 削除機能の作成	93
3.10. 更新機能の作成(登録機能の修正)	98
3.11. 注文番号一覧設定機能の作成	101
3.12. 金額計算機能の作成	107
3.13. 伝票発行機能の作成	112
4. コンポーネント接続関係(まとめ)	126

1. 概要

このチュートリアルでは、MySQL で作成した購買データベースと連携をとって購買伝票を発行するアプリケーションを作成します。アプリケーションの機能概要は以下の通りです。

なお、本書は MZ Platform の基本的な操作ができる方を対象に記載しております。コンポーネントの接続方法や起動メソッドの設定方法等は「開発チュートリアル（基礎編）」または「アプリケーションビルダー操作説明書」をご参照ください。また、MySQL の導入と設定につきましては、付属の「工程管理システム導入マニュアル」をご参照ください。

メインウィンドウ

MySQL データベースを操作するための機能を提供します。

- データベースへの接続
- データベースの新規作成
- テーブルの作成/更新/削除
- テーブルの一覧表示
- 選択されたテーブルの構成表示
- SQL 文実行機能
- データベーステーブルへの複数レコード一括追加機能
- データベースのバックアップ機能 ※ [データベース] メニューから実行
- データベーステーブル編集機能 ※ [データベース] メニューから実行

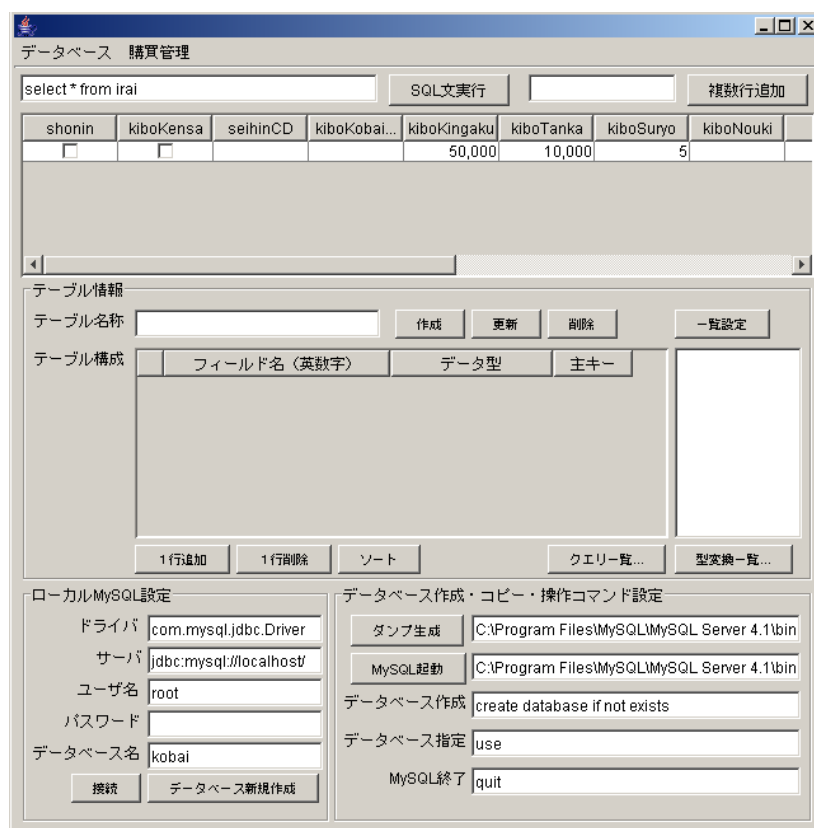


図 1 メインウィンドウ概観

発注依頼フレーム

発注依頼機能を提供します。メインウィンドウの[購買管理] - [発注依頼...]メニューを実行すると発注依頼フレームが表示されます。

- irai テーブルの検索
- irai テーブルへのレコード登録/削除/更新
- 発注依頼の伝票発行機能

注文番号	注文日	発注部門コード	購買品コード	発注数量	単価	金額	部門承認
A1	2005/12/20	S1	P1	5	10,000	50,000	☐

図 2 発注依頼フレーム概観

発注依頼書

注文番号	発注部門コード	購買品コード	注文日	部門承認	単価	発注数量	金額
A1	S1	P1	2005/12/20 00:00:00	FALSE	10000.0	5	50000

図 3 伝票発行

項目	主キー	注文番号	注文日	発注部門コード	購買品コード	メーカー名
フィールド名	id	kobaiCD	iraibi	bumonCD	hinmokuCD	maker
データ型	Integer	String	Date	String	String	String

規格仕様	希望納期	発注数量	単価	金額
kikaku	kiboNouki	kiboSuryo	kiboTanka	kiboKingaku
String	Date	Integer	Double	Double

(仮)注文先コード	製品設備コード	検査の有無	部門承認
kiboKobaisakiCD	seihinCD	kiboKensa	shonin
String	String	Boolean	Boolean

図 4 irai (発注依頼) テーブル構成図

2. データベース連携部の作成

まずは、データベースとの連携を行うデータベース連携部を作成します。ここでは、付属の「工程管理システム簡易版」で使用されている「ローカル MySQL 管理」複合コンポーネントを流用することになります。この複合コンポーネントは、「データベースアクセス」コンポーネントや「外部プログラム通信」コンポーネントなどから構成され、MySQL データベースを操作するための機能を提供します。

操 作

ProductionControl.mzax をロードしましょう。

- ① アプリケーションビルダーを起動します。
- ② **ロード** ボタンをクリックし「AP_DATA¥Production¥ProductionControl.mzax」をロードします。

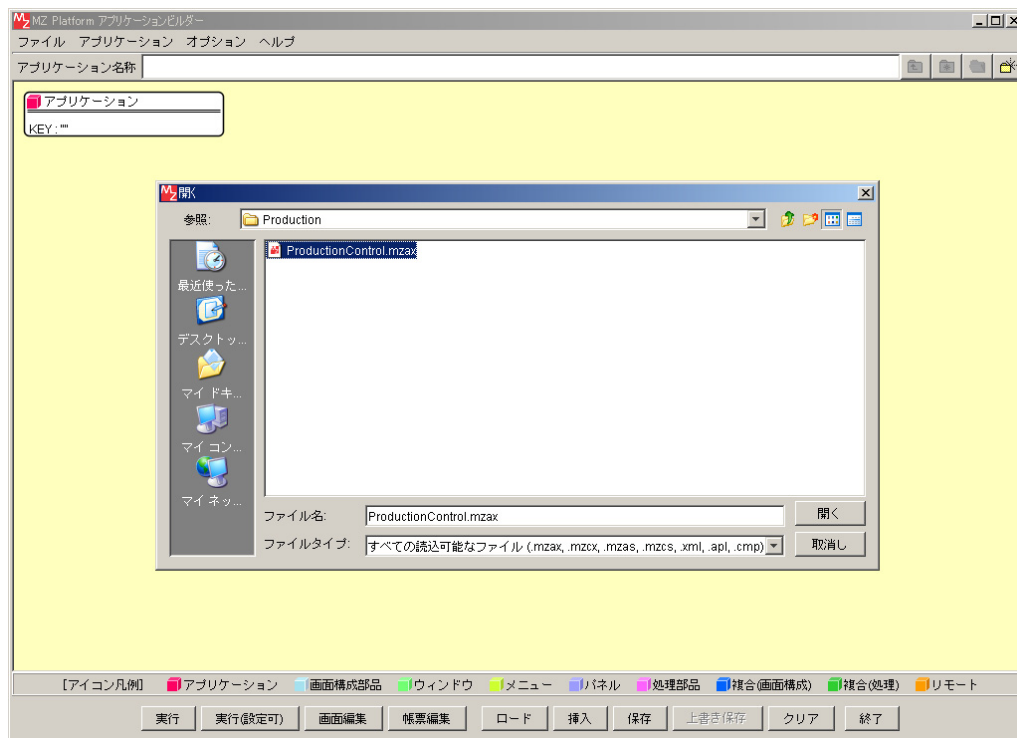


図 5 ProductionControl.mzax のロード

2.1. 「ローカルMySQL管理」複合コンポーネントの保存

「ローカル MySQL 管理」複合コンポーネントを流用するために、これをいったん別ファイルとして保存しておきます。

操 作

「ローカル MySQL 管理」複合コンポーネントを保存しましょう。

- ① 「階層選択」アイコンをクリックし、[アプリケーション階層選択] ダイアログを表示します。
- ② 「ローカル MySQL 管理」へ移動します。

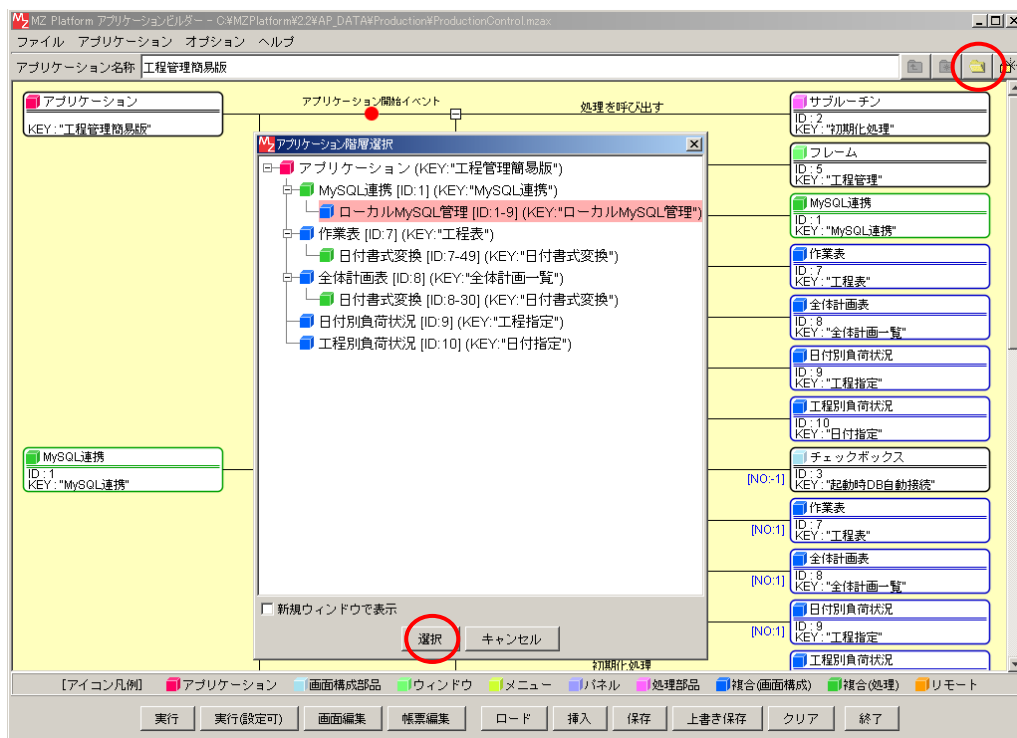


図 6 「ローカル MySQL 管理」への移動

③ 複合コンポーネント「ローカル MySQL 管理」を保存します。

保存 ボタンをクリックし、表示された [確認] ダイアログで **この階層** のみを選択します。

ファイル名を「MySQL.mzcx」として保存します。拡張子「.mzcx」は、保存時に自動的に付加されます。

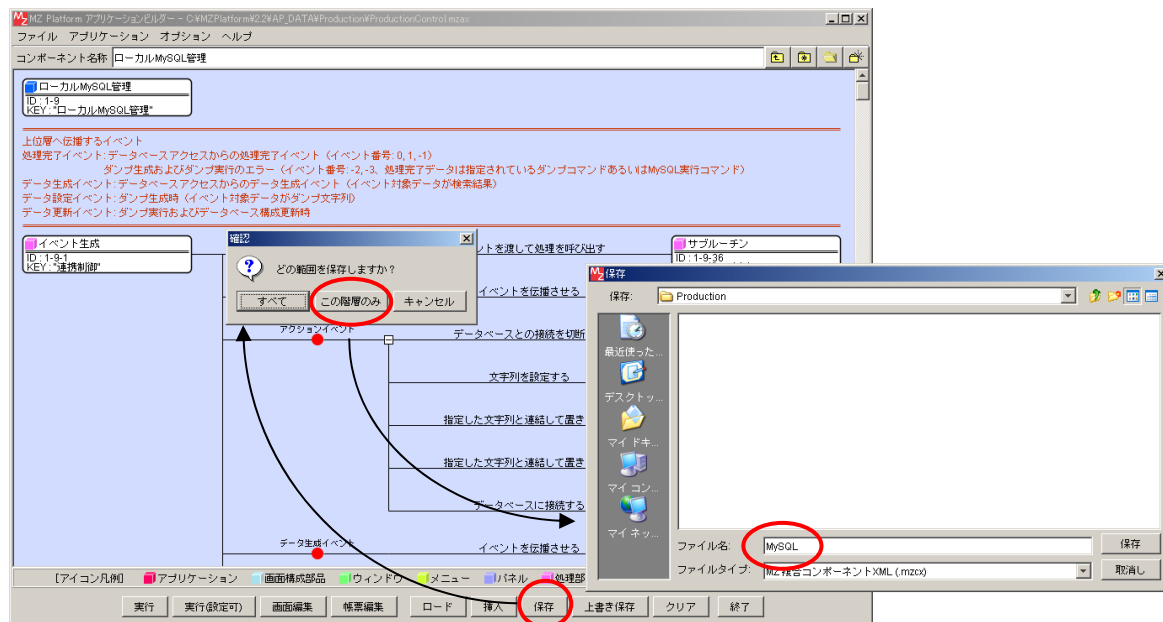


図 7 複合コンポーネント「ローカル MySQL 管理」の保存

④ アプリケーションビルダーのトップ画面に移動し、**クリア** ボタンをクリックして、画面をクリアします。

2.2. 「ローカルMySQL管理」複合コンポーネントのロード

アプリケーションビルダトップ画面に「フレーム」コンポーネントを配置し、アプリケーション起動時に「フレーム」が表示されるように接続します。

操 作 コンポーネントを追加し、接続しましょう。

- ① 必要なコンポーネントを追加します。
作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [ウィンドウ] - [フレーム] とクリックします。
- ② 下の表に示すようにコンポーネントを接続します。

開始

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ アプリケーション
発生イベント	アプリケーション開始イベント
接続先コンポーネント	■ フレーム (ID:1)
起動メソッド	フレームを表示する()

終了

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ フレーム (ID:1)
発生イベント	アクションイベント
接続先コンポーネント	■ アプリケーション
起動メソッド	アプリケーションを終了する()

確認

接続関係と配置は以下のようになります。

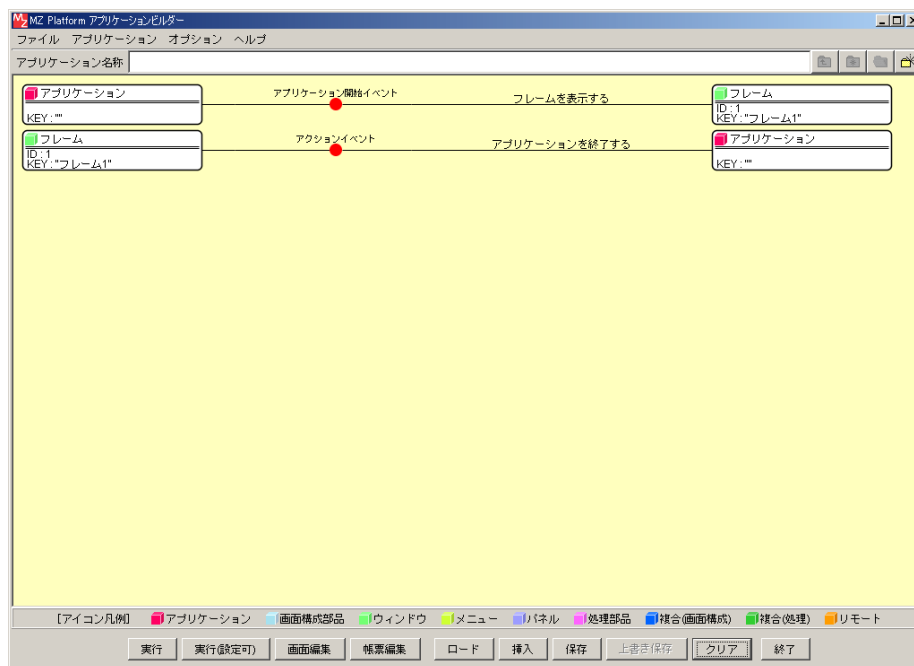


図 8 フレームコンポーネントの配置

操作

「ローカル MySQL 管理」複合コンポーネントを追加し画面に配置しましょう。

- ① 前節でファイルに保存した「ローカル MySQL 管理」複合コンポーネント「MySQL.mzcX」をロードします。アプリケーションビルダー画面上で
右クリック - [複合コンポーネント追加] - [ロード...] とクリックします。
- ② 表示された[開く]ダイアログから、「MySQL.mzcX」を選択し、**開く**ボタンをクリックします。

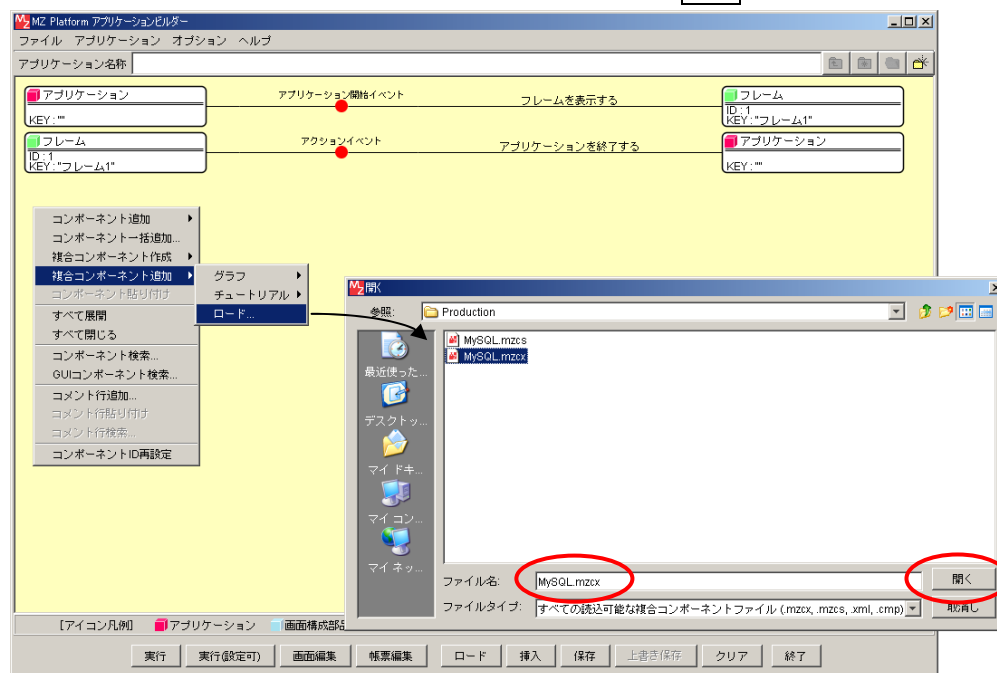


図 9 「ローカル MySQL 管理」複合コンポーネントのロード

確認

接続関係と配置は以下のようになります。

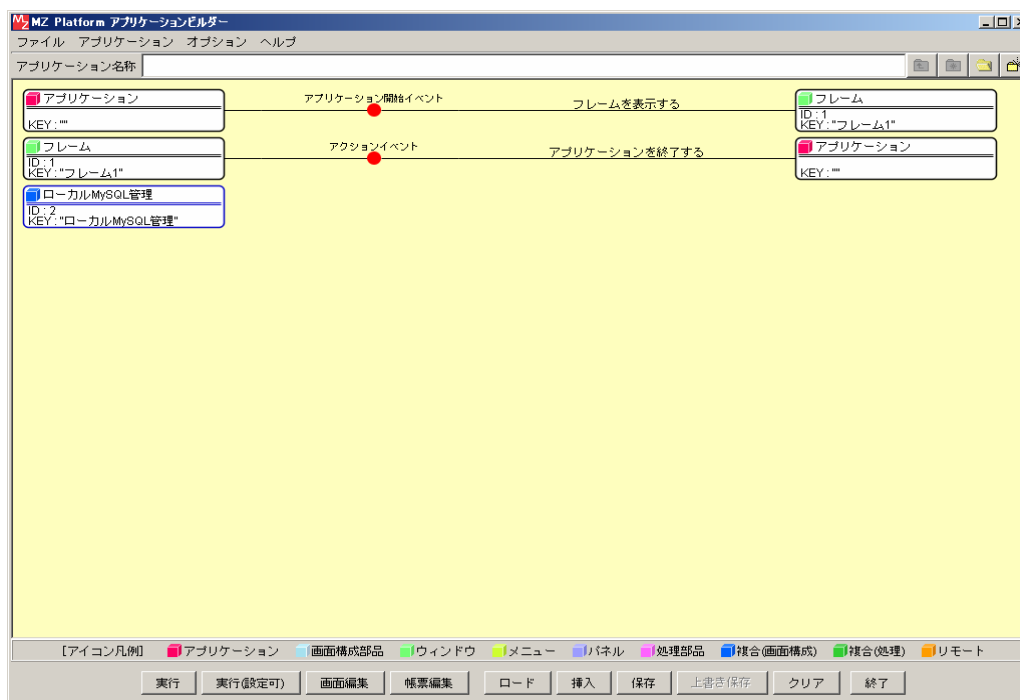


図 10 「ローカル MySQL 管理」複合コンポーネントの配置

- ③ **画面編集**ボタンをクリックし、画面編集画面で「ローカル MySQL 管理」複合コンポーネントをフレームに追加します。

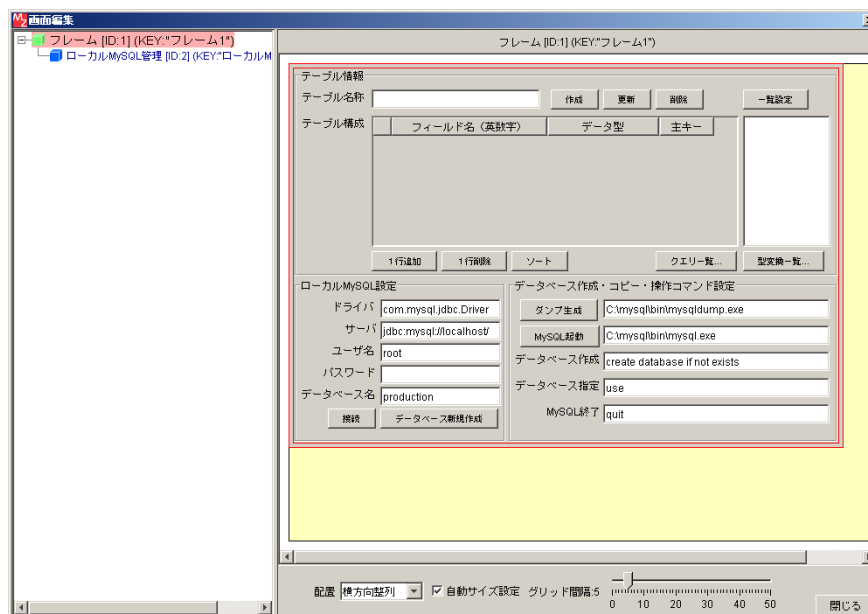


図 11 「ローカル MySQL 管理」複合コンポーネントのフレームへの追加

- ④ **閉じる**ボタンをクリックして、アプリケーションビルダー画面に戻ります。

- ⑤ **実行**もしくは**実行(設定可)**ボタンをクリックして、アプリケーションが正しく起動することを確認します¹。確認後はアプリケーションを終了します。

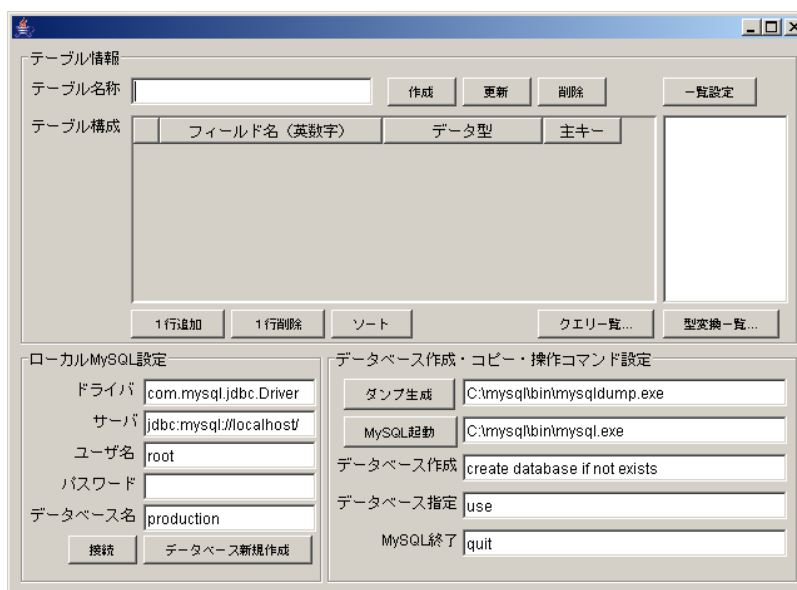


図 12 アプリケーション起動画面

操作 「購買管理.mzax」として保存しましょう。

- ① **保存**ボタンをクリックします。
- ② 保存先フォルダを「AP_DATA」にし、ファイル名を「購買管理.mzax」として保存します。

¹ 購買管理 01.mzax

2.3. 「ローカルMySQL管理」複合コンポーネントの設定

【1】初期化处理と終了処理の設定

アプリケーション開始時に初期化处理を、終了時に終了処理を行うようにイベント接続を行います。

操 作 アプリケーションの初期化处理と終了処理を設定しましょう。

- ① 下の表に示すようにコンポーネントを接続します。

開始処理

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ アプリケーション
発生イベント	アプリケーション開始イベント
接続先コンポーネント	■ ローカル MySQL 管理 (ID:2)
起動メソッド	初期化处理()

終了処理

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ アプリケーション
発生イベント	アプリケーション終了イベント
接続先コンポーネント	■ ローカル MySQL 管理 (ID:2)
起動メソッド	終了処理()

- ② アプリケーション開始イベントから接続されている起動メソッドの順番を入れ替えます。
「フレーム」コンポーネントを「ローカルMySQL管理」複合コンポーネントの下に配置します。

確 認 接続関係と配置は以下のようになります。

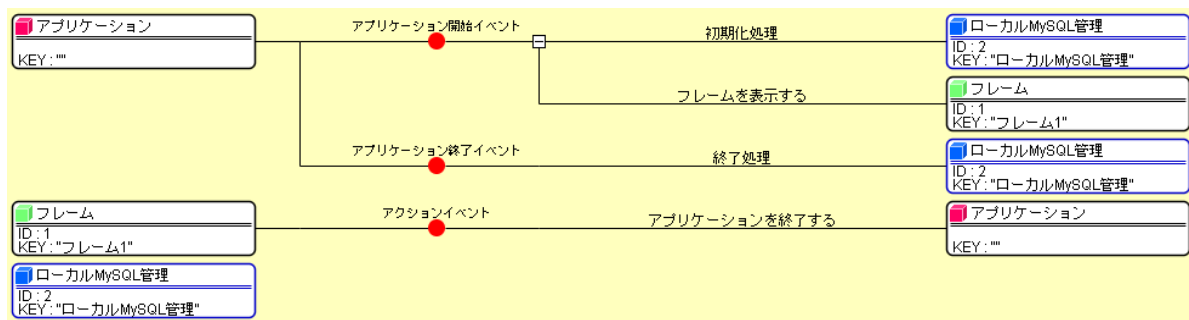


図 13 初期化处理と終了処理の設定

【2】MySQL ダンプ生成コマンドのパス設定

次に、MySQL の mysqldump コマンドのパスを設定します。使用する MySQL のバージョンに合わせてパス設定します。

操作 MySQL の mysqldump コマンドのパスを設定しましょう。

- ① **実行**もしくは**実行(設定可)**ボタンをクリックして、アプリケーションを起動します。
- ② [ダンプ生成]ボタンをクリックします。
表示された [開く] ダイアログで、MySQL のダンプ生成コマンドのパスを設定します。
MySQL を標準インストールした場合、そのパスは以下のようになります。

MySQL 4.0.x の場合: C:\mysql\bin\mysqldump.exe

MySQL 4.1.x の場合: C:\Program Files\MySQL\MySQL Server 4.1\bin\mysqldump.exe

MySQL 5.0.x の場合: C:\Program Files\MySQL\MySQL Server 5.0\bin\mysqldump.exe

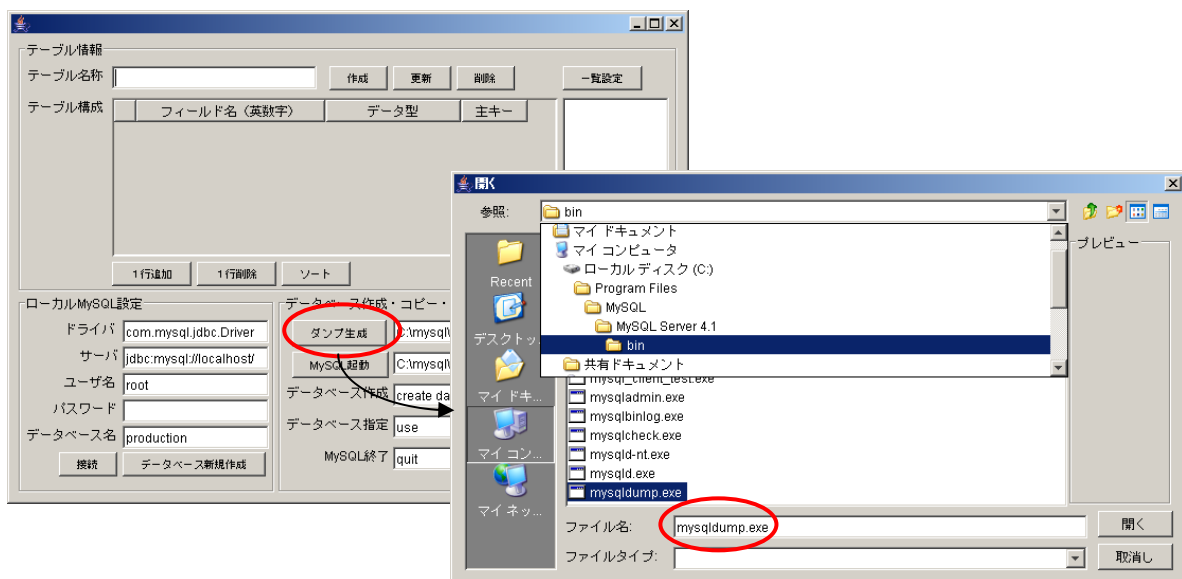


図 14 MySQL ダンプ生成コマンドパスの設定

ワンポイント - ダンプ -

MySQL のデータは mysqldump というツールを使用することで、テキストファイルに出力することができます。ファイルの中身は MySQL でそのまま実行できる SQL 文が記述されているため、作成したダンプファイルを利用することで、データベースのバックアップと復元が可能となります。

【3】MySQL 起動コマンドのパス設定

MySQL の起動コマンドのパスを設定します。使用する MySQL のバージョンに合わせてパスを設定します。

操 作 MySQL の起動コマンドのパスを設定しましょう。

- ① [MySQL 起動] ボタンをクリックします。
- ② 表示された[開く]ダイアログで、MySQL の起動コマンドのパスを設定します。
MySQL を標準インストールした場合、そのパスは以下のようになります。

MySQL 4.0.x の場合: C:\mysql\bin\mysql.exe

MySQL 4.1.x の場合: C:\Program Files\MySQL\MySQL Server 4.1\bin\mysql.exe

MySQL 5.0.x の場合: C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe

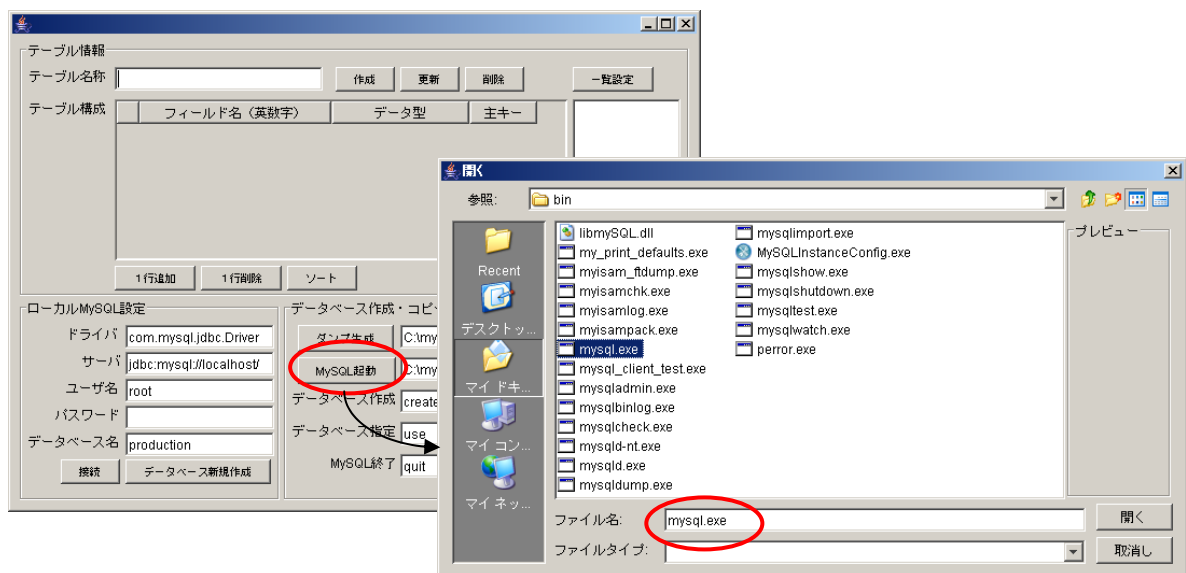


図 15 MySQL 起動コマンドの設定

【4】型変換テーブルおよび「データベースアクセス」コンポーネントの設定

(MySQL 4.1.0 以降を使用する場合)

MySQL 4.0.x 以前では論理型 (BOOLEAN) のフィールドがサポート外でしたので、「ローカル MySQL 管理」複合コンポーネントでは、Java の論理型 (java.lang.Boolean) のデータを enum('true','false') に変換して扱っています。MySQL 4.1.0 からは BOOLEAN がサポートされるようになりましたので、MySQL 4.1.0 以降を使用する場合には、それに合わせて型変換テーブルと「データベースアクセス」コンポーネントの設定を直しておきます。もちろん、元のままでも動作に差し支えはありません。

操 作 型変換テーブルの設定を変更しましょう。

- ① [型変換一覧...] ボタンをクリックし、型変換一覧ダイアログを表示させます。
- ② java.lang.Boolean に対応する MySQL のデータ型名を「BOOLEAN」に書き換え、ダイアログを閉じます。

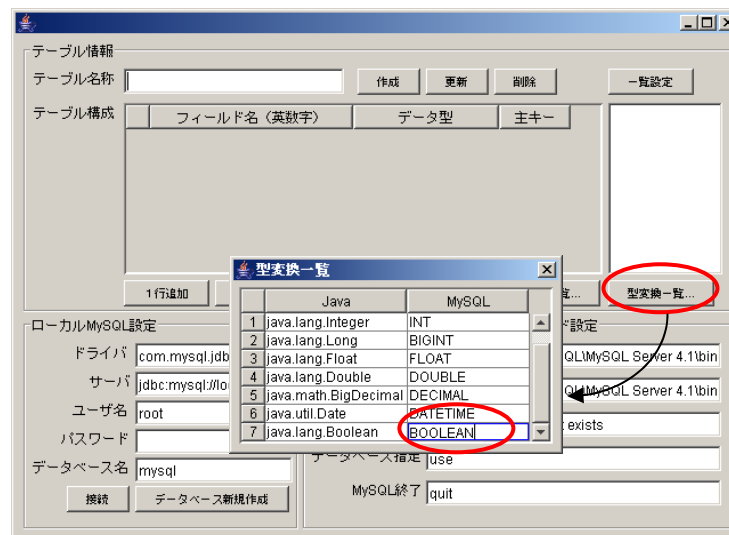


図 16 型変換テーブルの修正

- ③ アプリケーションを終了します。

操 作

「データベースアクセス」コンポーネントの型変換設定を確認しましょう。

- ① 「ローカル MySQL 管理」複合コンポーネントへ移動します。
- ② 「ID2-4 KEY. ” ローカル MySQL アクセス” 」の「データベース」コンポーネントの属性情報画面を表示します。Java の論理型(`java.lang.Boolean`)のデータを `enum('true', 'false')` に変換する場合には[EnumBooleanMapping]を[true]を、BOOLEAN として扱う場合には[false]を選択し、[×] (閉じる) ボタンをクリックします。

The screenshot shows a dialog box titled 'M2 コンポーネント属性情報' (M2 Component Attribute Information). It contains a table of properties for the 'ローカルMySQLアクセス' (Local MySQL Access) component. The 'EnumBooleanMapping' property is highlighted with a red circle, and the 'false' radio button is selected. The 'ComponentKeys' property shows '日本語: ローカルMySQLアクセス' and '英語: ' (English:).

ComponentKey	ローカルMySQLアクセス	☐ NULL
ComponentID	4	
ComponentPublicName		☒ NULL
AllowRemoteInvocation	☐ true ☒ false	
AllowPullTransfer	☐ true ☒ false	
AllowPushTransfer	☐ true ☒ false	
ComponentKeys	日本語: ローカルMySQLアクセス	英語: ☐ NULL
EnumBooleanMapping	☐ true ☒ false	

設定

図 17 データベースアクセスコンポーネントの型変換設定確認

ワンポイント — コンポーネント検索 —

コンポーネント検索機能を使用すると、目的のコンポーネントを一覧から検索し、選択できるので便利です。

コンポーネントの検索画面を表示するには、アプリケーションビルダー画面上で右クリック— [コンポーネント検索...] をクリックします。

【5】ユーザ名、パスワード、データベース名の設定

ユーザ名とパスワードを、ご自身の MySQL の環境に合わせて設定します。

操 作

ユーザ名、パスワード、データベース名を設定しましょう。

- ① アプリケーションビルダーのトップ画面に移動します。
- ② **実行**もしくは**実行(設定可)**ボタンをクリックして、アプリケーションを起動します。
- ③ ユーザ名とパスワードを、ご自身の MySQL の環境に合わせて設定します。
- ④ データベース名に、さしあたり「mysql」と入力します。

2.4. 動作確認

ここで、アプリケーションを少し操作してみることにします。各ボタンの機能は下図の通りです²。



図 18 アプリケーションの動作

操作 アプリケーションの動作を確認しましょう。

- ① [接続]ボタンをクリックし、データベースに接続します。
この場合、接続先は「mysql」データベースです。
- ② [一覧設定]ボタンをクリックします。
右上に、「mysql」データベースに含まれるテーブルの一覧が表示されます。
- ③ テーブル一覧の中から、任意のテーブル名を選択します。
そのテーブルのフィールド名、データ型、主キー設定の有無が左側のテーブルに表示されます。

² 購買管理 02.mzax

このアプリケーションからは、データベースやテーブルの作成を行うこともできます。接続先データベースを「test」に変更し、実際に新規に「machine」テーブルを作成してみます。

操 作 「test」データベースに接続しましょう。

- ① データベース名に「test」と入力します。
- ② [接続]ボタンをクリックします。

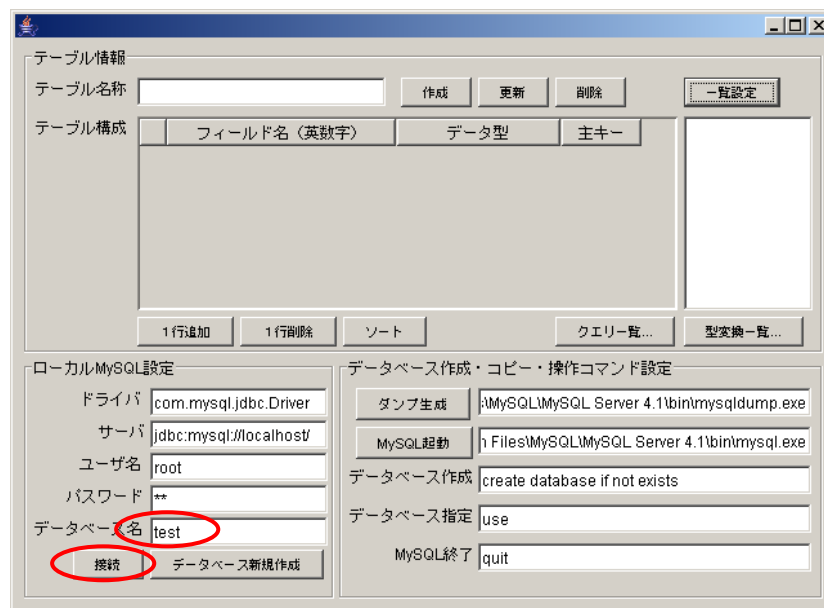


図 19 接続先データベースの変更

- ③ [一覧設定]ボタンをクリックします。
「test」データベースに含まれるテーブルの一覧が表示されます。

ここでは、「test」データベースに以下のようなテーブルを作成することにします。

テーブル名: machine

フィールド名、データ型、主キー属性:

フィールド名	id	name	entryDay
データ型	整数(Integer)	文字列(String)	日付(Date)
主キー属性	主キー(true)	非主キー(false)	非主キー(false)

!!! 注意事項 !!!

「ローカル MySQL 管理」複合コンポーネントでは、主キーを設定できるフィールドは1つで、そのデータ型は整数(Integer、Long)に限定されます。また、属性 auto_increment が設定されます。

操作 「machine」テーブルを作成しましょう。

- ① テーブル名称に「machine」と入力します。

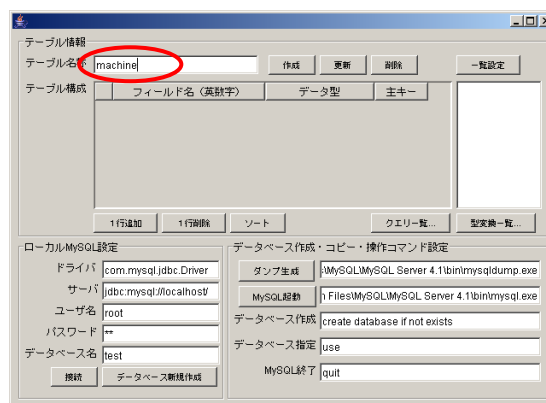


図 20 テーブル名称の設定

- ② [1行追加]ボタンをクリックします。
表に行が追加されます。

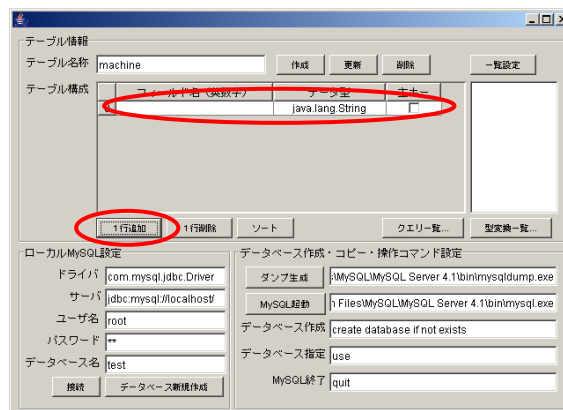


図 21 行の追加

- ③ 最初のフィールドを設定します。フィールド名「id」を入力します。データ型「java.lang.Integer」をプルダウンリストから選択します。主キー属性は「true」のため、チェックボックスにチェックを入れます。

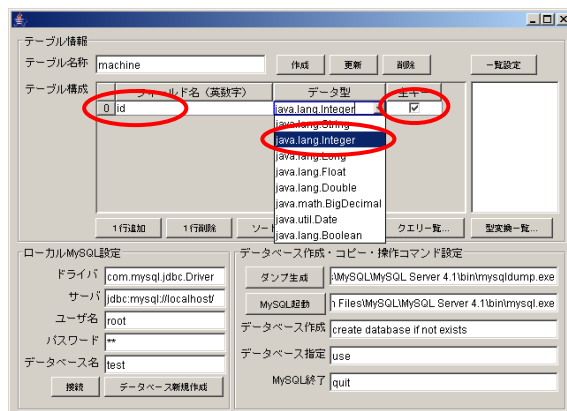


図 22 フィールドの設定(id)

- ④ 2つ目のフィールドを設定します。[1行追加]ボタンをクリックします。
先頭に行が新たに追加されます。
フィールド名は「name」、データ型は「java.lang.String」、主キー属性は「false」とします。

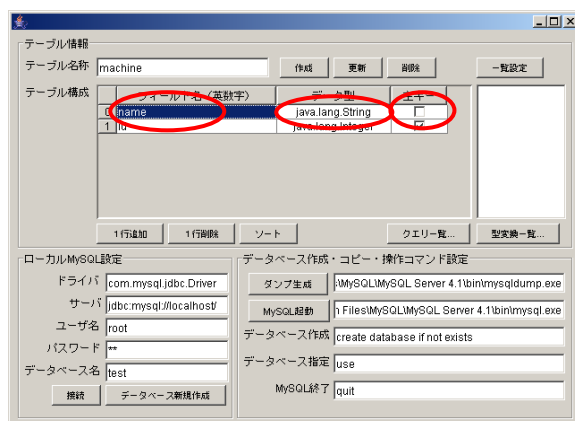


図 23 フィールドの設定(name)

- ⑤ 残りの1つのフィールドを設定します。[1行追加]ボタンをクリックします。
先頭に行が新たに追加されます。
フィールド名は「entryDay」、データ型は「java.util.Date」、主キー属性は「false」とします。

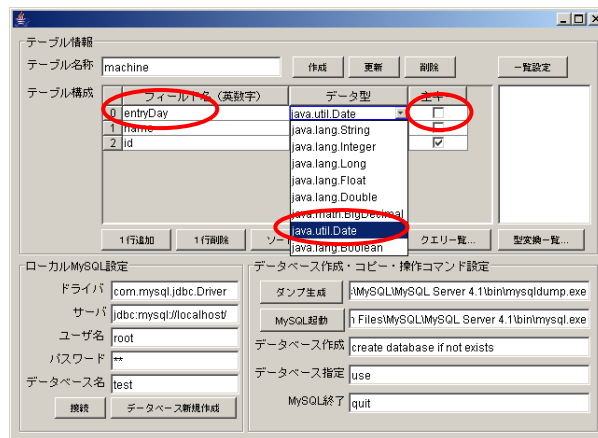


図 24 フィールドの設定(entryDay)

- ⑥ [作成]ボタンをクリックします。
テーブルが作成され、一覧の最後尾にその名前が表示されます。

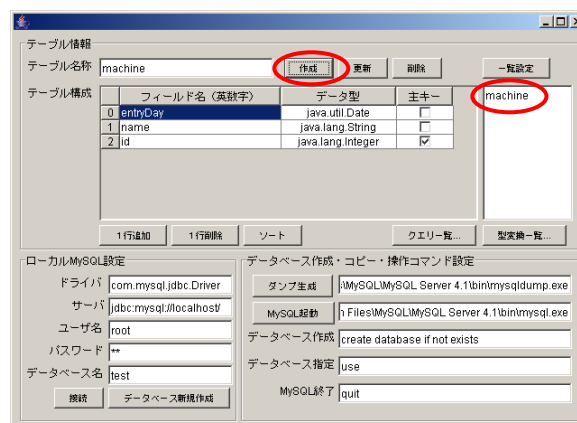


図 25 テーブルの作成

MySQL Command Line Client を起動し、実際にテーブルが作成されていることを確認します。

操 作 テーブルが作成されていることを確認しましょう。

- ① MySQL Command Line Client を起動します。

MySQL 4.0.x を使用する場合:

[スタート] - [(すべての)プログラム] - [アクセサリ] - [コマンドプロンプト] とクリックし、次のコマンドを実行します。

```
C:> %mysql%bin\mysql -u root -p
```

MySQL 4.1.x を使用する場合:

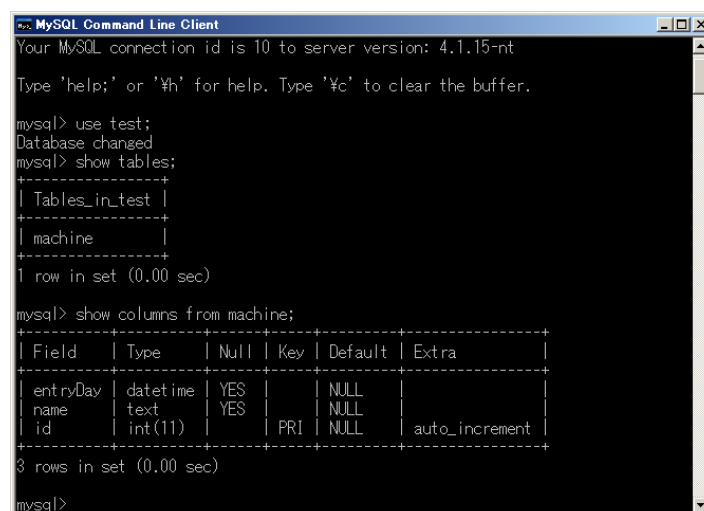
[スタート] - [プログラム] - [MySQL] - [MySQL Sever 4.1] - [MySQL Command Line Client] とクリックします。

MySQL 5.0.x を使用する場合:

[スタート] - [プログラム] - [MySQL] - [MySQL Sever 4.0] - [MySQL Command Line Client] とクリックします。

- ② パスワードを入力後、次のようにコマンドを実行して確認します。

```
mysql> use test;
mysql> show tables;
mysql> show columns from machine;
```



```
MySQL Command Line Client
Your MySQL connection id is 10 to server version: 4.1.15-nt
Type 'help;' or '?' for help. Type '\c' to clear the buffer.

mysql> use test;
Database changed
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| machine        |
+-----+
1 row in set (0.00 sec)

mysql> show columns from machine;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| entryDay | datetime | YES |  | NULL |  |
| name    | text    | YES |  | NULL |  |
| id      | int(11) | YES | PRI | NULL | auto_increment |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

図 26 MySQL Command Line Client によるテーブル作成の確認

- ③ 確認後、MySQL Command Line Client を終了します。次のようにコマンドを実行します。

```
mysql> quit;
```

- ④ アプリケーションを終了します。

2.5. SQL文実行機能の追加

ここからは、「ローカル MySQL 管理」複合コンポーネントのメソッドを利用して、様々な機能を追加していくことにします。「ローカル MySQL 管理」複合コンポーネントの主な公開メソッドおよび発生するイベントは以下の通りです。

公開メソッド

メソッド	説明
初期化処理()	この複合コンポーネントを使用するための準備を行います。 <u>使用前に必ず起動してください。</u>
終了処理()	この複合コンポーネントの使用を終えるために必要な処理を行います。 <u>アプリケーション終了時に必ず起動してください。</u>
ローカル MySQL の接続・切断(int)	MySQL データベースシステムへの接続と切断を行います。引数に 0 を与えると切断、1 を与えると接続になります。
SQL 文を実行する(String)	与えられた SQL 文を実行します。検索などのように、処理結果が返ってくる SQL 文を実行した場合、SQL 文実行結果は、この複合コンポーネントが発生するデータ生成イベントの内包データとして取得できます。
イベント番号を指定して SQL 文を実行する(String, int)	与えられた SQL 文を実行します。指定した整数は、SQL 文実行結果が含まれるデータ生成イベントのイベント番号になります。
指定したテーブルに複数行を一括追加する(String, PFObjectTable)	String で指定したテーブルに、PFObjectTable に記述された複数の行(レコード)を追加します。
テーブルに記述した SQL 文を実行する(PFObjectTable)	PFObjectTable に記述された複数の SQL 文を順次実行します。
ダンプ生成()	接続先データベースのダンプを生成します。生成されたダンプは、この複合コンポーネントが発生するデータ設定イベントの内包データとして取得できます。
ダンプ実行(Object, int)	object に与えたダンプ(データベースを作成する一群の SQL 文)を実行し、データベースの内容を更新します。 <u>int は使われていません。</u>

発生するイベント

- 処理完了イベント

発生のタイミング	対象データ	イベント番号
データベースへの接続および切断	0	0
SQL 文エラーの発生	0	-1
実行結果を戻さない SQL 文の実行	更新された行数	0 もしくは指定した番号

- データ生成イベント

発生のタイミング	対象データ	イベント番号
実行結果を戻す SQL 文の実行	SQL 文実行結果	0 もしくは指定した番号

- データ設定イベント

発生のタイミング	対象データ	イベント番号
ダンプ生成()の実行	データベースのダンプ	0

- データ更新イベント

発生のタイミング	対象データ	更新データ	イベント番号
ダンプ実行 (Object, int) の実行 によるデータベース内容の更新	MySQL コマンドのフ ルパス名	更新したデータ ベース名	0

まずは、SQL文の実行機能を追加します。³

追加するコンポーネント

- テキストフィールド
- ボタン
- テーブル

追加する機能

1. ボタンをクリックしたら、
2. テキストフィールドに記入されている SQL 文をイベント番号 1 を指定して実行し、
3. 実行結果をテーブルに設定する。

操 作 コンポーネントを追加し、接続しましょう。

- ① 必要なコンポーネントを追加します。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [テキスト] - [テキストフィールド] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [ボタン] - [ボタン] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [テーブル] - [テーブル] とクリックします。

- ② 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。

[テキストフィールド] コンポーネントの[Columns]を「25」、[ComponentKey]を「SQL 文」と設定します。

[ボタン] コンポーネントの[text]を「SQL 文実行」と設定します。

³ 購買管理 03.mzax

③ 下の表に示すようにコンポーネントを接続します。

テキストフィールドに記入されている SQL 文をイベント番号 1 を指定して実行する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ボタン (ID:4, KEY:SQL 文実行)
発生イベント	アクションイベント
接続先コンポーネント	■ ローカル MySQL 管理 (ID:2)
起動メソッド	イベント番号を指定して SQL 文を実行する (String, int)
<引数 0>	取得方法: メソッド戻り値 コンポーネント: テキストフィールド (ID:3, KEY:SQL 文) メソッド/値: テキストを取得する
<引数 1>	取得方法: 固定値 メソッド/値: 1

実行結果をテーブルに設定する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ローカル MySQL 管理 (ID:2)
発生イベント	データ生成イベント
接続先コンポーネント	■ テーブル (ID:5)
起動メソッド	テーブルデータを設定する (PFObjectTable)
<引数>	説明: テーブルデータ 取得方法: イベント内包 メソッド/値: イベント対象データ
イベント番号	1

SQL 文実行結果は、「ローカル MySQL 管理」複合コンポーネントが発生するデータ生成イベントの内包データとして取得します。SQL 文実行時にイベント番号として 1 を指定しましたので、データ生成イベントのイベント番号も 1 となることにご注意ください。

確認

接続関係と配置は以下ようになります。

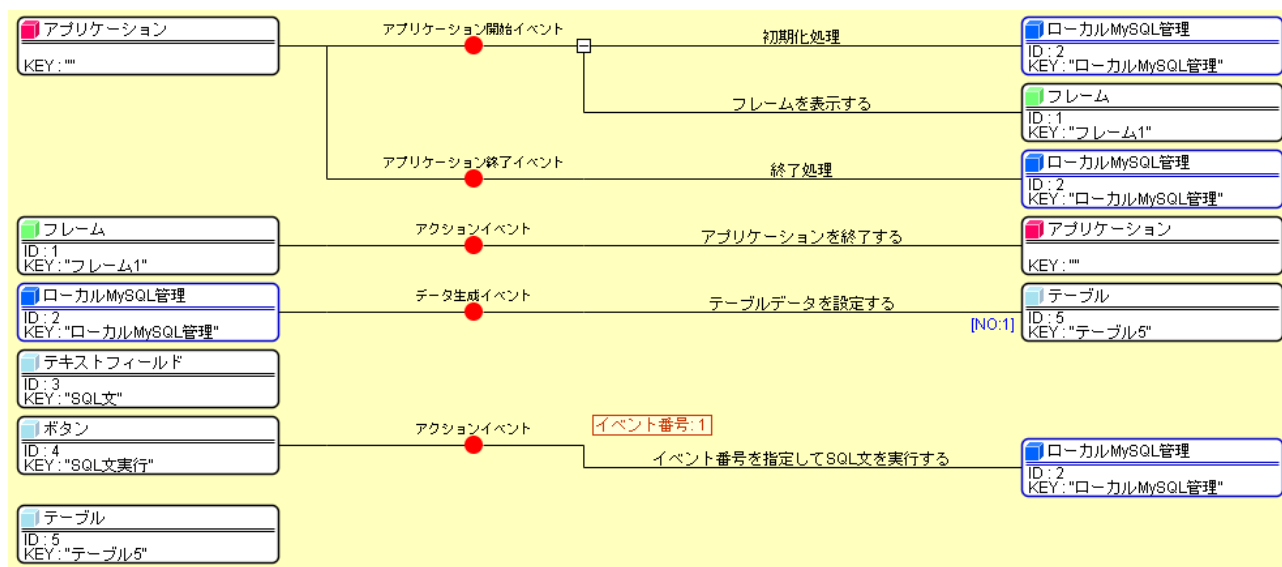


図 27 SQL 文実行機能の追加(コンポーネント接続関係)

ワンポイント コメント

指定したイベント番号が分かるように、イベント番号をコメントとして記述しておくとき便利です。起動メソッドにコメントを記述するには、コメントを記述する接続先コンポーネントを右クリック → [起動メソッドコメント編集...] とクリックします。

操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 配置方法を「手動配置」に切り替えます。
- ③ 画面編集画面で「テキストフィールド」コンポーネントをフレームに追加します。
- ④ 画面編集画面で「ボタン」コンポーネントをフレームに追加します。
- ⑤ 画面編集画面で「テーブル」コンポーネントをフレームに追加します。

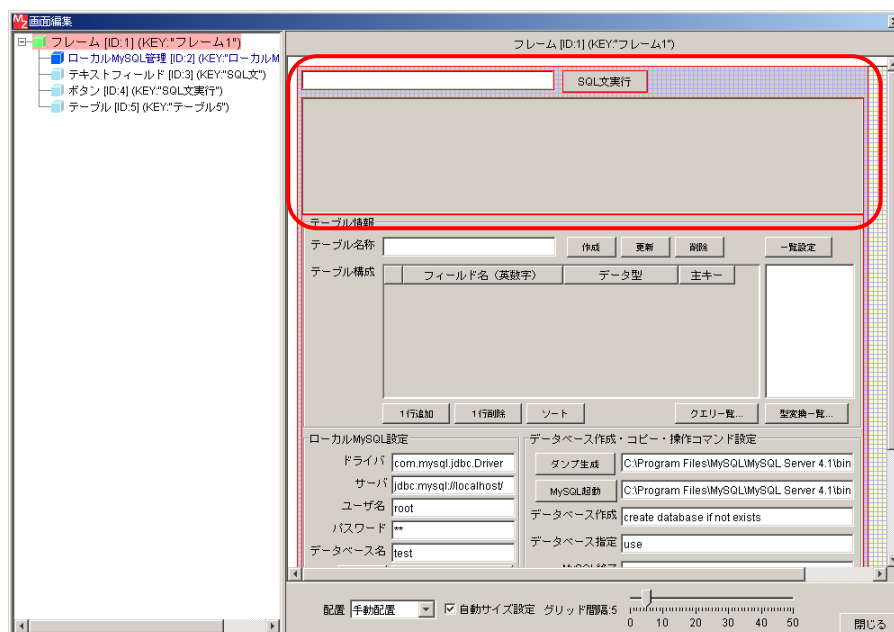


図 28 SQL 文実行機能の追加(コンポーネントの画面配置)

- ⑥ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

操 作

アプリケーションの動作を確認しましょう。

- ① [実行]もしくは[実行(設定可)]ボタンをクリックし、アプリケーションを起動します。
- ② [接続]ボタンをクリックし、「test」データベースに接続します。
- ③ [一覧設定]ボタンをクリックし、テーブル一覧の中に「machine」テーブルが含まれていることを確認します。
- ④ テキストフィールドに「select * from machine」とSQL文を入力します。
- ⑤ [SQL文実行]ボタンをクリックし、行データが空のテーブルが表示されることを確認します。

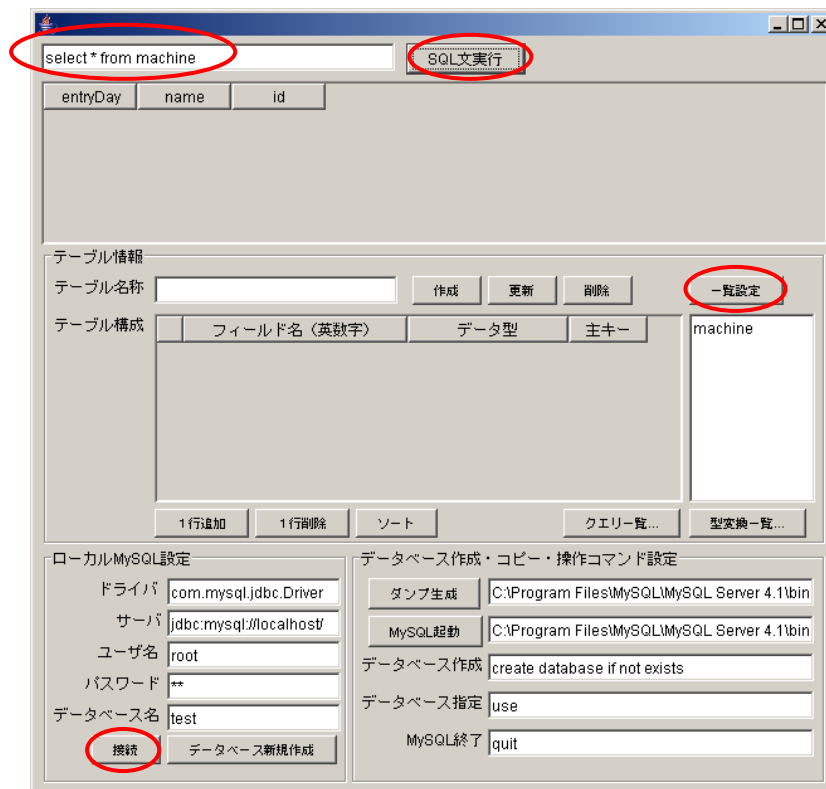


図 29 SQL 文の実行

- ⑥ 確認後、アプリケーションを終了します。

2.6. 複数行一括追加機能の追加

データベーステーブルに複数のレコードを一括して追加する機能を追加します。⁴

追加するコンポーネント

- テキストフィールド
- ボタン

追加する機能

1. ボタンをクリックしたら、
2. テキストフィールドで指定したデータベーステーブルに対して
3. テーブルに記入されている複数の行データを一括追加する。

操 作	コンポーネントを追加し、接続しましょう。
------------	----------------------

- ① 必要なコンポーネントを追加します。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [テキスト] - [テキストフィールド] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [ボタン] - [ボタン] とクリックします。

- ② 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。

[ボタン] コンポーネントの[text]を「複数行追加」と設定します。

[テキストフィールド] コンポーネントの[ComponentKey]を「追加先テーブル名」と設定します。

⁴ 購買管理 04.mzax

③ 下の表に示すようにコンポーネントを接続します。

指定したテーブルに複数の行データを一括追加する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ボタン (ID:7, KEY:複数行追加)
発生イベント	アクションイベント
接続先コンポーネント	■ ローカル MySQL 管理 (ID:2)
起動メソッド	指定したテーブルに複数行を一括追加する (String, PFObjectTable)
<引数 0>	取得方法: メソッド戻り値 コンポーネント: テキストフィールド (ID:6, KEY: 追加先テーブル名) メソッド/値: テキストを取得する
<引数 1>	取得方法: メソッド戻り値 コンポーネント: テーブル (ID:5) メソッド/値: テーブルデータを取得する

確認 接続関係と配置は以下のようになります。

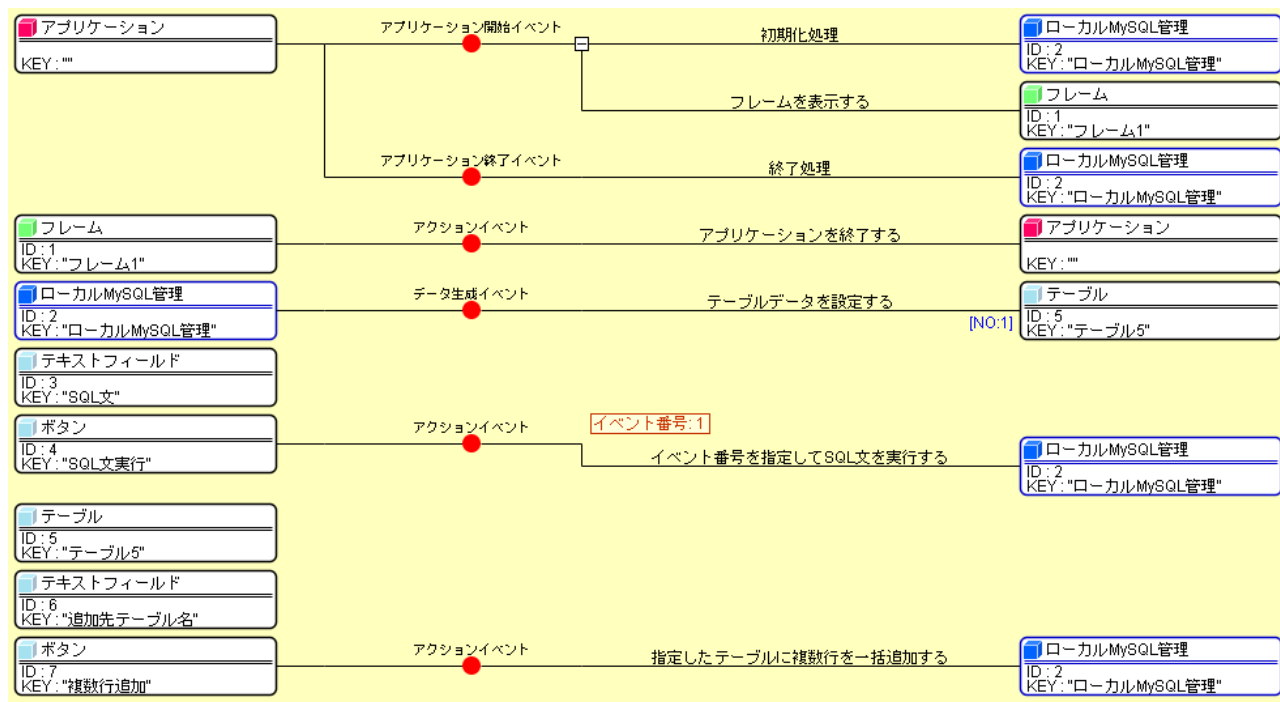


図 30 複数行一括追加機能の追加 (コンポーネント接続関係)

操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 画面編集画面で「テキストフィールド」コンポーネントをフレームに追加します。
- ③ 画面編集画面で「ボタン」コンポーネントをフレームに追加します。

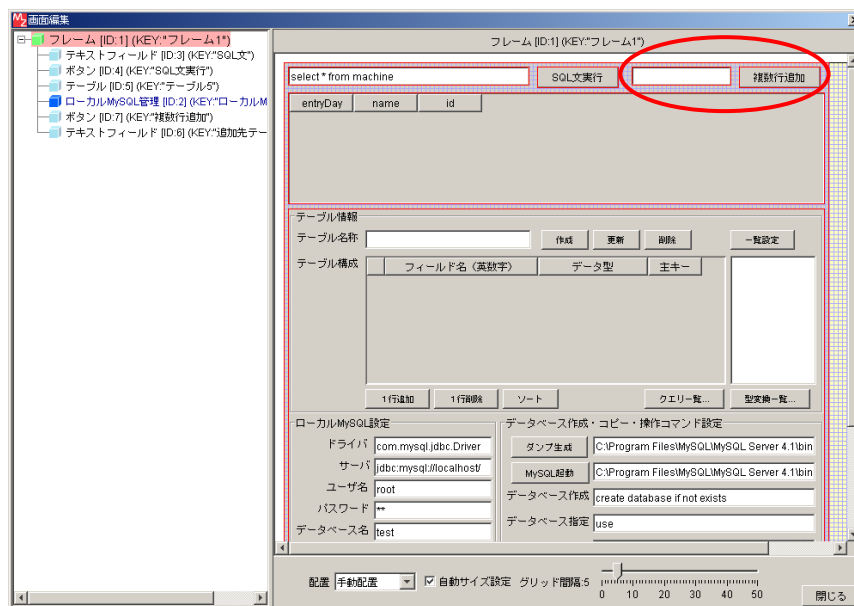


図 31 複数行一括追加機能の追加(コンポーネントの画面配置)

- ④ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

操 作

アプリケーションの動作を確認しましょう。

- ① **実行(設定可)**ボタンをクリックし、アプリケーションを起動します。
- ② **[接続]**ボタンをクリックし、「test」データベースに接続します。
- ③ テーブルにレコードを追加します。
テーブル上で右クリック-[追加]-[行]-[複数行...]をクリックし、3行と設定します。
- ④ テーブルにデータを入力します。
「entryDay」と「name」に任意にデータを入力します。
「id」は自動設定されますので、入力はありません。
- ⑤ テーブル名として「machine」を設定し、[複数行追加]ボタンをクリックします。

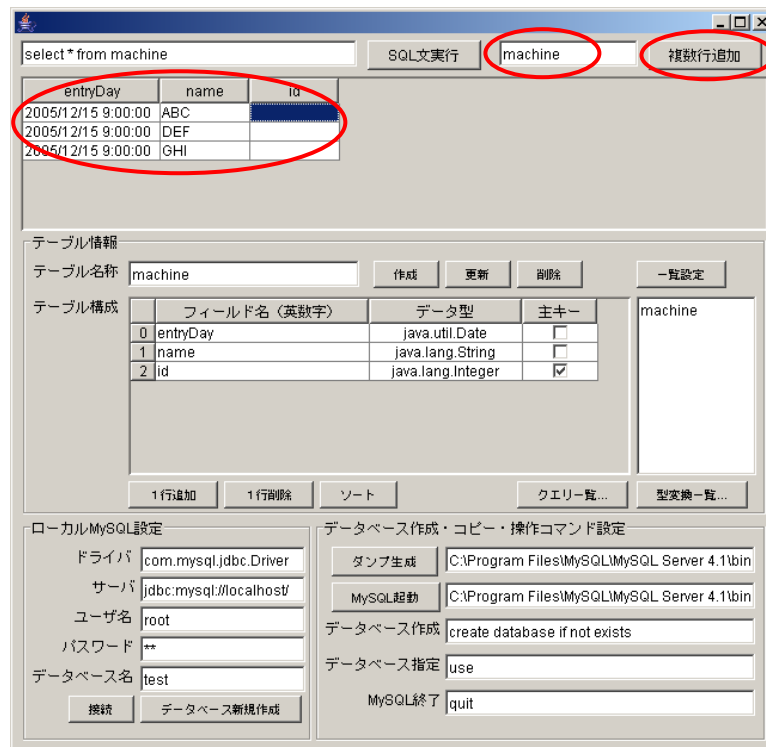


図 32 複数行一括追加

- ⑥ [SQL 文実行]ボタンをクリックし、結果を確認します。
id が自動で設定されていることが確認できます。

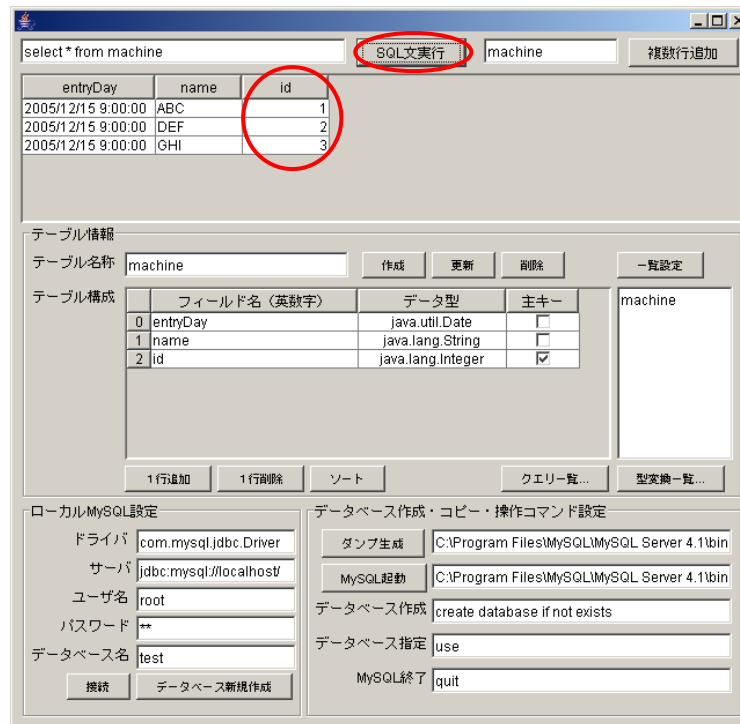


図 33 複数行追加結果の確認

- ⑦ アプリケーションを終了します。

2.7. バックアップ機能の追加

データベースのバックアップには、MySQLのダンプ機能を利用します。⁵

追加するコンポーネント

- メニュー
- メニューアイテム
- ファイル選択
- テキストファイル出力

追加する機能

1. バックアップメニューを選択したら、
2. 出力先ファイルを選択し、
3. ダンプ文字列を書き出す。
4. ダンプ文字列を書き出した後、出力先ファイルを閉じる。

操 作	コンポーネントを追加し、接続しましょう。
------------	----------------------

- ① 必要なコンポーネントを追加します。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [メニュー] - [メニュー] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [メニュー] - [メニューアイテム] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [ダイアログ] - [ファイル選択] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [入出力] - [ファイル] - [テキストファイル出力] とクリックします。

- ② 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。

[メニュー] コンポーネントの[text]を「データベース」、

[メニューアイテム] コンポーネントの[text]を「バックアップ...」、

[ファイル選択] コンポーネントの[ComponentKey]を「ダンプ出力ファイル選択」、

[テキストファイル出力] コンポーネントの[ComponentKey]を「ダンプファイル出力」と設定します。

⁵ 購買管理 05.mzax

③ 下の表に示すようにコンポーネントを接続します。

バックアップメニューを選択したらファイル選択ダイアログを表示する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■メニューアイテム (ID:9, KEY:バックアップ)
発生イベント	アクションイベント
接続先コンポーネント	■ファイル選択 (ID:10, KEY:ダンプ出力ファイル選択)
起動メソッド	単数 Save 用ファイル選択ダイアログを表示する (Component)
<引数>	説明: 親コンポーネント 取得方法: 固定値 メソッド/値: なし

選択したファイルを設定する ①

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ファイル選択 (ID:10, KEY:ダンプ出力ファイル選択)
発生イベント	データ選択イベント
接続先コンポーネント ①	■テキストファイル出力 (ID:11, KEY:ダンプファイル出力)
起動メソッド	setFile(File)
<引数>	取得方法: イベント内包 メソッド/値: イベント対象データ
イベント番号	1

ダンプファイルを生成する ②

接続先コンポーネント ②	■ローカル MySQL 管理 (ID:2)
起動メソッド	ダンプ生成()
イベント番号	1

ファイルを指定して、ダンプ文字列を書き出す①

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ローカル MySQL 管理 (ID:2)
発生イベント	データ設定イベント
接続先コンポーネント①	■ テキストファイル出力 (ID:11, KEY:ダンプファイル出力)
起動メソッド	ファイルオブジェクトを指定して文字列を書き出す (File, String)
<引数 0>	説明: ファイルオブジェクト 取得方法: メソッド戻り値 コンポーネント: テキストファイル出力 (ID:11, KEY:ダンプファイル出力) メソッド/値: ファイルオブジェクトを取得する
<引数 1>	説明: 書き出す文字列 取得方法: イベント内包 メソッド/値: イベント対象データ

ファイルを閉じる②

接続先コンポーネント②	■ テキストファイル出力 (ID:11, KEY:ダンプファイル出力)
起動メソッド	ファイルを閉じる()

確認

接続関係と配置は以下ようになります。

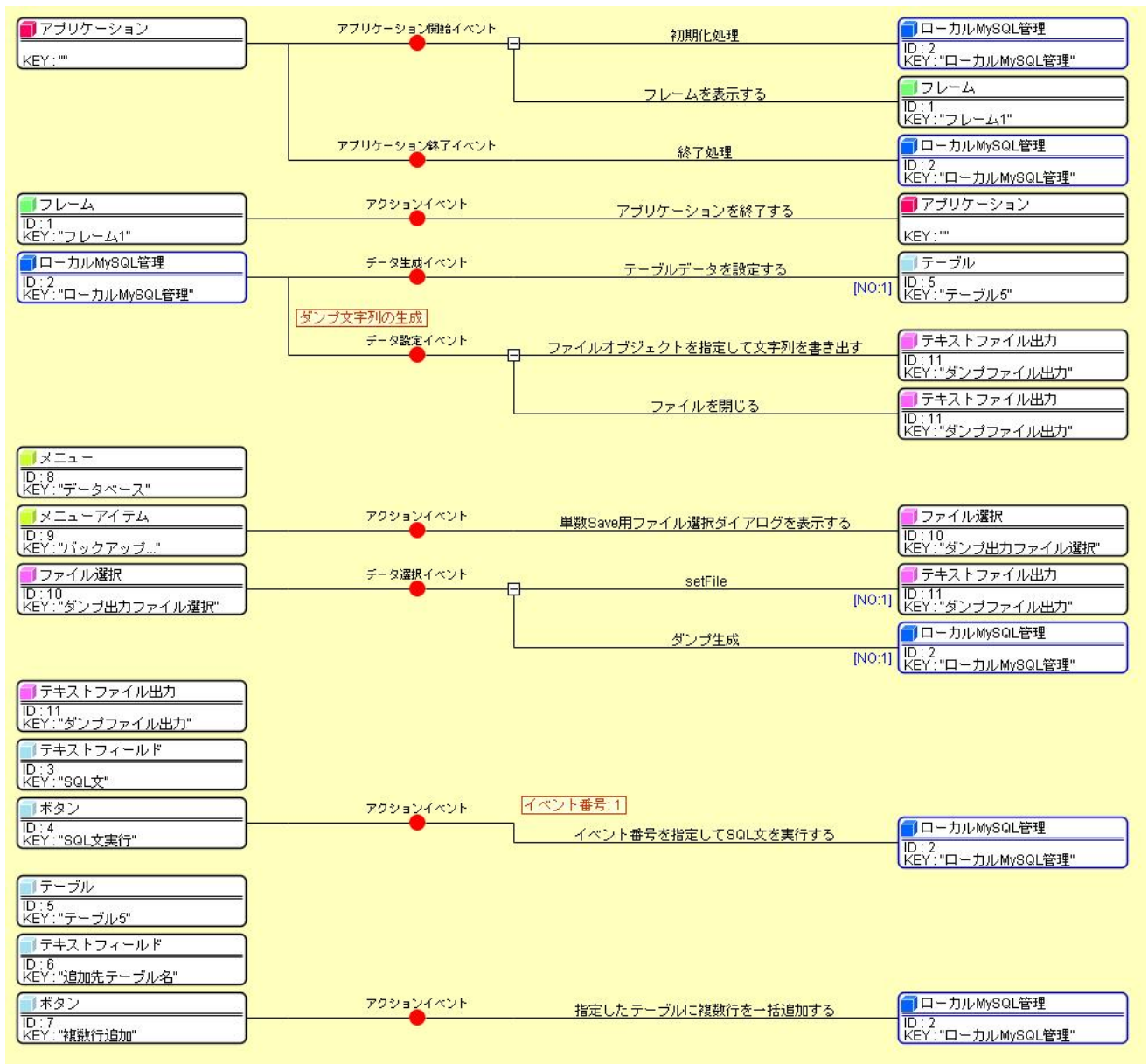


図 34 バックアップ機能の追加(コンポーネント接続関係)

操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 「メニュー」コンポーネントをフレームに追加します。左側のコンポーネント一覧から「フレーム」コンポーネントを右クリック－[コンポーネント追加]－[メニュー] とクリックします。
- ③ 「メニューアイテム」コンポーネントをメニューに追加します。左側のコンポーネント一覧から「メニュー」コンポーネントを右クリック－[コンポーネント追加]－[メニューアイテム] とクリックします。
- ④ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

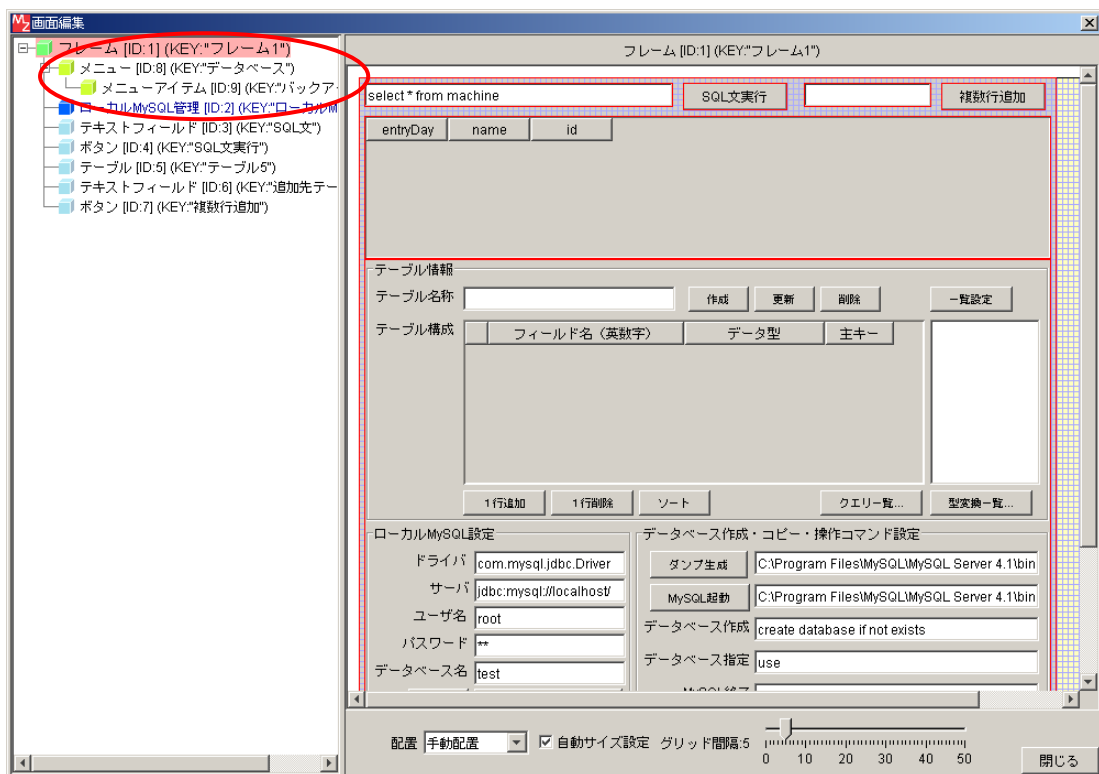


図 35 バックアップ機能の追加(メニューの追加)

操 作

アプリケーションの動作を確認しましょう。

- ① [実行]もしくは[実行(設定可)]ボタンをクリックし、アプリケーションを起動します。
- ② [接続]ボタンをクリックし、「test」データベースに接続します。
- ③ メニューから[データベース]-[バックアップ...]と選択し、バックアップファイルを作成します。
保存先フォルダを「AP_DATA¥Production」にし、ファイル名を「backup.sql」として保存します。

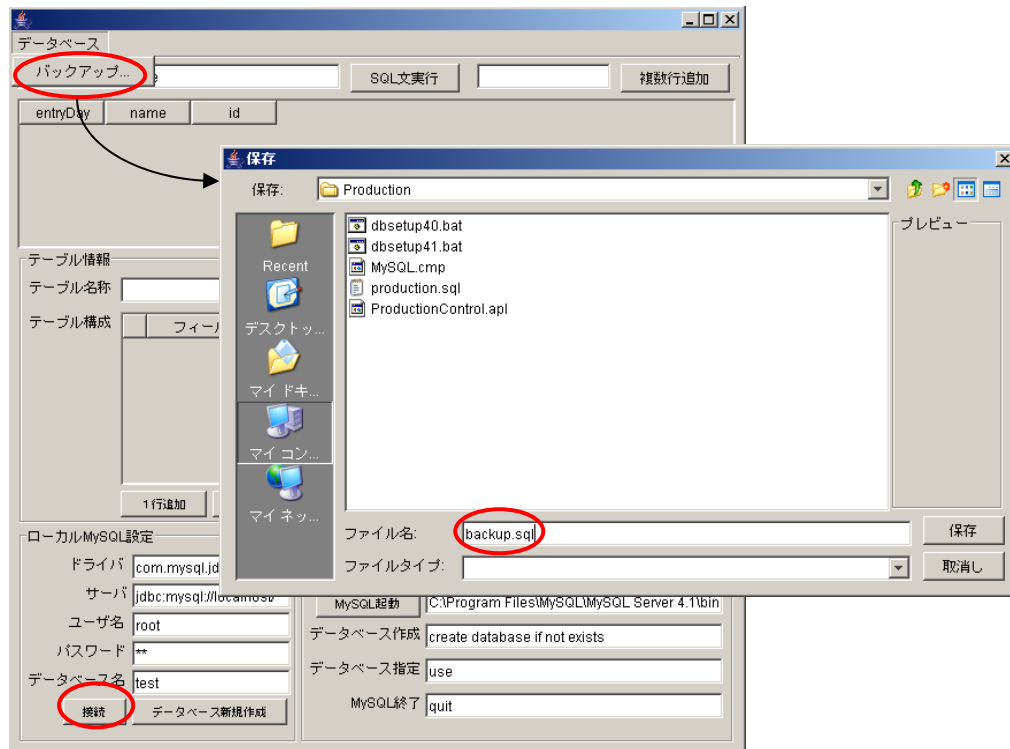


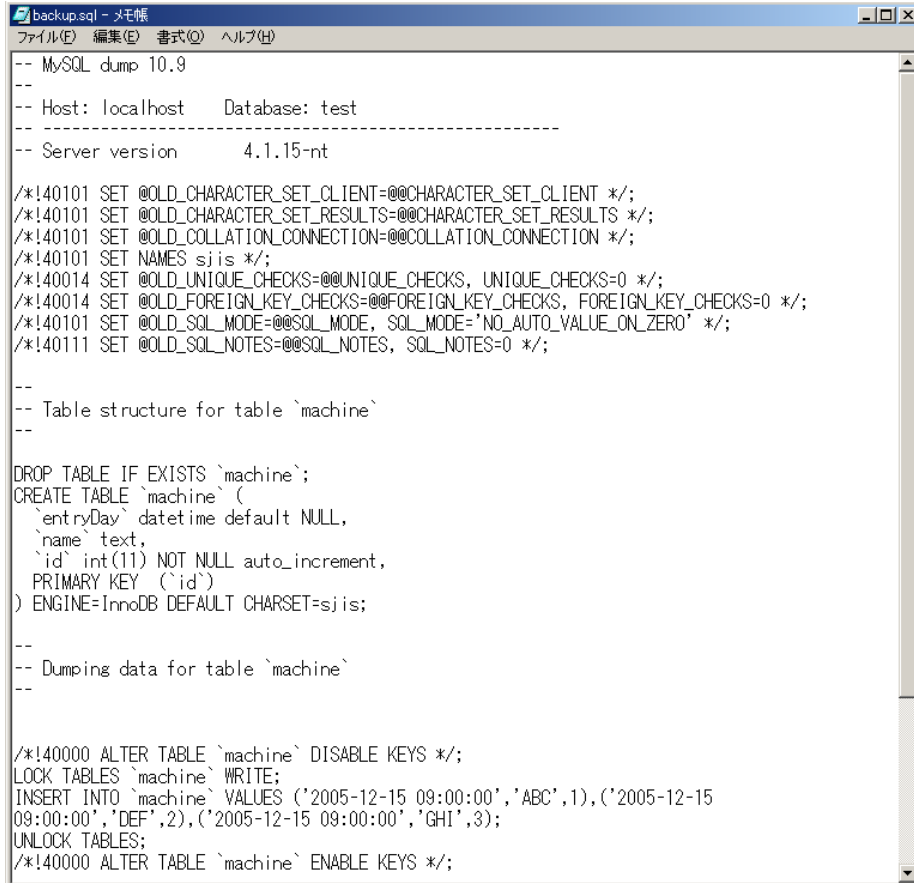
図 36 バックアップファイルの作成(ダンプ出力)

- ④ アプリケーションを終了します。

操 作

出力先ファイル(backup.sql)の内容を確認しましょう。

- ① エクスプローラを起動し、「backup.sql」ファイルを保存したフォルダを表示します。
- ② 「backup.sql」ファイルをダブルクリックし、ファイルを開きます。
データベースを作成するための一群の SQL 文が記述されていることが確認できます。



```
-- MySQL dump 10.9
--
-- Host: localhost    Database: test
--
-- Server version 4.1.15-nt
--
-- Table structure for table `machine`
--
DROP TABLE IF EXISTS `machine`;
CREATE TABLE `machine` (
  `entryDay` datetime default NULL,
  `name` text,
  `id` int(11) NOT NULL auto_increment,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=sjis;
--
-- Dumping data for table `machine`
--

/*!40000 ALTER TABLE `machine` DISABLE KEYS */;
LOCK TABLES `machine` WRITE;
INSERT INTO `machine` VALUES ('2005-12-15 09:00:00','ABC',1),('2005-12-15
09:00:00','DEF',2),('2005-12-15 09:00:00','GHI',3);
UNLOCK TABLES;
/*!40000 ALTER TABLE `machine` ENABLE KEYS */;
```

図 37 backup.sql

- ③ 確認後、ファイルを閉じます。

2.8. データベース内容設定機能の追加

作成したバックアップファイルから、データベースの内容を設定する機能を追加します。⁶

追加するコンポーネント

- メニューアイテム
- ファイル選択
- テキストファイル入力

追加する機能

1. 内容設定メニューを選択したら、
2. バックアップファイルを選択し、
3. データベースの内容を設定する。
4. ダンプ実行後、バックアップファイルを閉じる。

操 作	コンポーネントを追加し、接続しましょう。
-----	----------------------

- ① 必要なコンポーネントを追加します。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [メニュー] - [メニューアイテム] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [ダイアログ] - [ファイル選択] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [入出力] - [ファイル] - [テキストファイル入力] とクリックします。

- ② 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。

[メニューアイテム] コンポーネントの[text]を「内容設定...」、

[ファイル選択] コンポーネントの[ComponentKey]を「バックアップファイル選択」、

[テキストファイル入力] コンポーネントの[ComponentKey]を「バックアップファイル入力」と設定します。

⁶ 購買管理 06.mzax

③ 下の表に示すようにコンポーネントを接続します。

内容設定メニューを選択したらファイル選択ダイアログを表示する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■メニューアイテム (ID:12, KEY:内容設定...)
発生イベント	アクションイベント
接続先コンポーネント	■ファイル選択 (ID:13, KEY:バックアップファイル選択)
起動メソッド	単数 Open 用ファイル選択ダイアログを表示する (Component)
<引数>	説明: 親コンポーネント 取得方法: 固定値 メソッド/値: なし

ファイルオブジェクトを指定して読み込む①

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ファイル選択 (ID:13, KEY:バックアップファイル選択)
発生イベント	データ選択イベント
接続先コンポーネント①	■テキストファイル入力 (ID:14, KEY:バックアップファイル入力)
起動メソッド	ファイルオブジェクトを指定して読み込む (File)
<引数>	説明: 読み込むファイルオブジェクト 取得方法: イベント内包 メソッド/値: イベント対象データ
イベント番号	1

ファイルを閉じる②

接続先コンポーネント②	■テキストファイル入力 (ID:14, KEY:バックアップファイル入力)
起動メソッド	ファイルを閉じる ()
イベント番号	1

データベースの内容を設定する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■テキストファイル入力 (ID:14, KEY:バックアップファイル入力)
発生イベント	データ生成イベント
接続先コンポーネント	■ローカルMySQL管理 (ID:2)
起動メソッド	ダンプ実行(Object, int)
<引数 0>	取得方法: イベント内包 メソッド/値: イベント対象データ
<引数 1>	取得方法: 固定値 メソッド/値: 0

確認

接続関係と配置は以下のようになります。

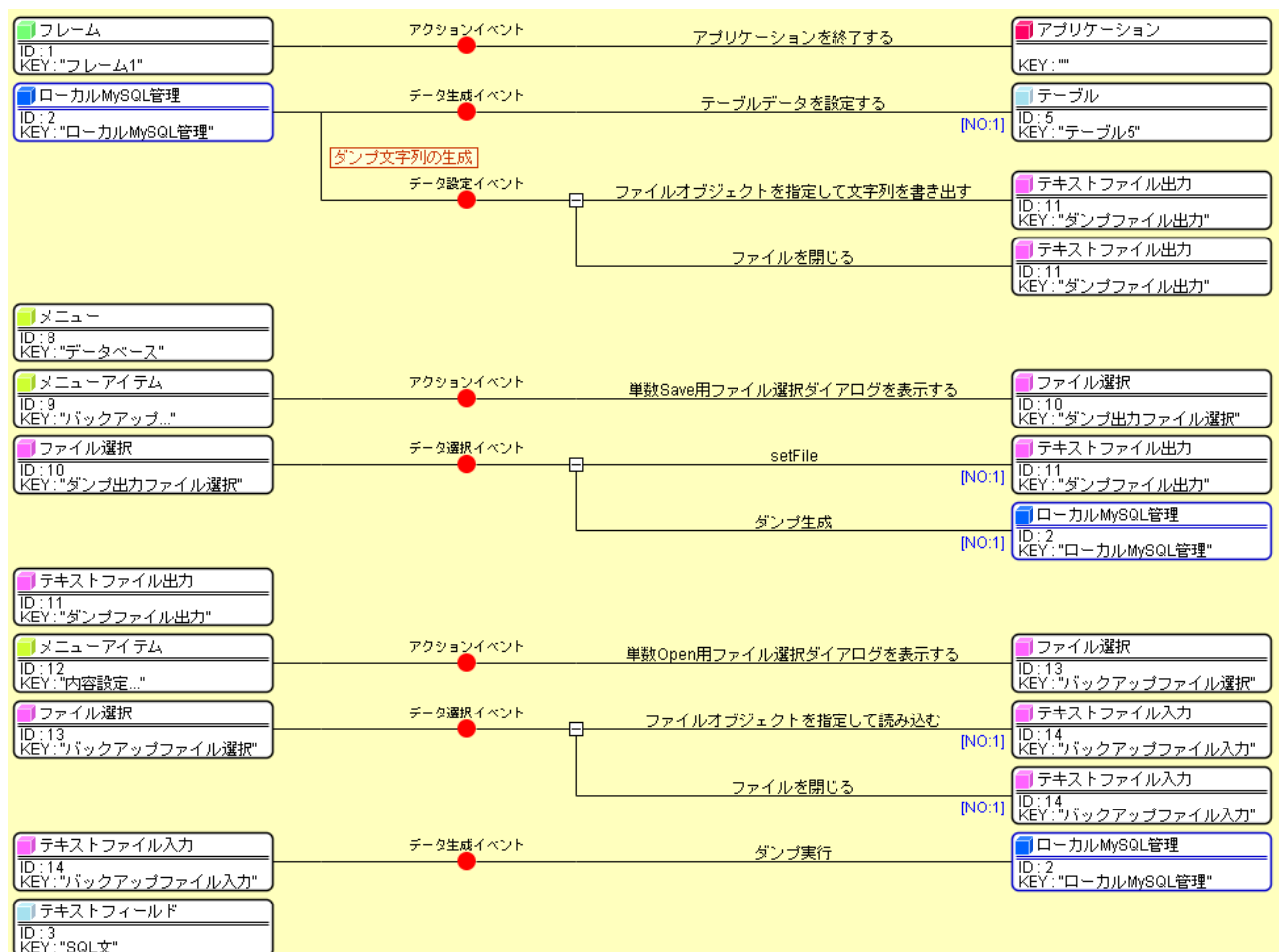


図 38 データベース内容設定機能の追加(コンポーネント接続関係)

操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 「メニューアイテム」コンポーネントをメニューに追加します。左側のコンポーネント一覧から「メニュー」コンポーネントを右クリック → [コンポーネント追加] → [メニューアイテム] とクリックします。
- ③ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

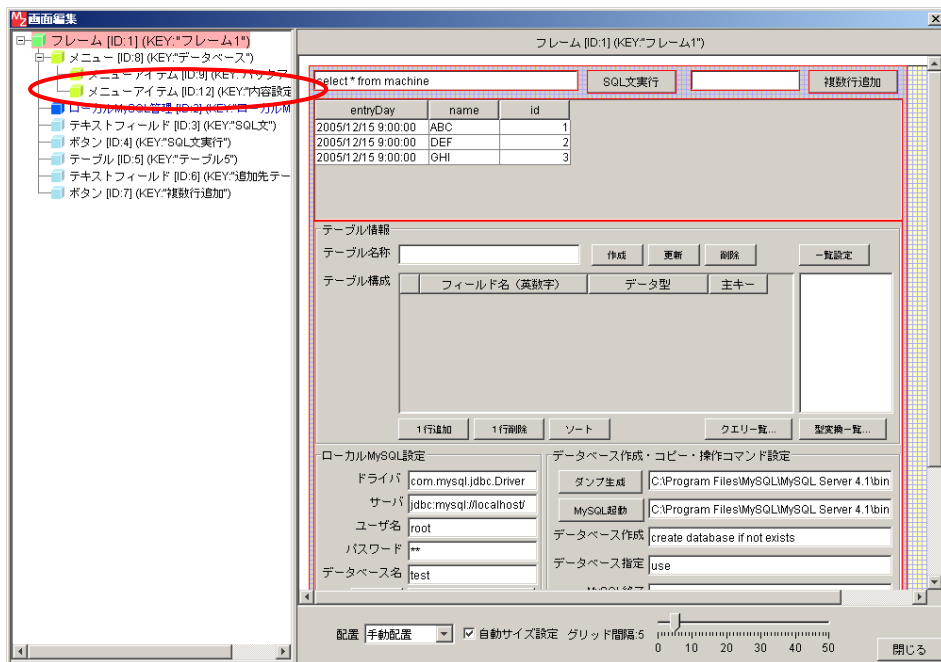


図 39 データベース内容設定機能の追加(メニューへの追加)

それでは、新しく「test1」という名前のデータベースを作成し、その内容を、前節で作成したバックアップファイルで設定してみます。

操 作

「test1」データベースを「backup.sql」を使って作成しましょう。

- ① **実行**もしくは**実行(設定可)**ボタンをクリックし、アプリケーションを起動します。
- ② [データベース名] に「test1」と入力し、[データベース新規作成]ボタンをクリックします。

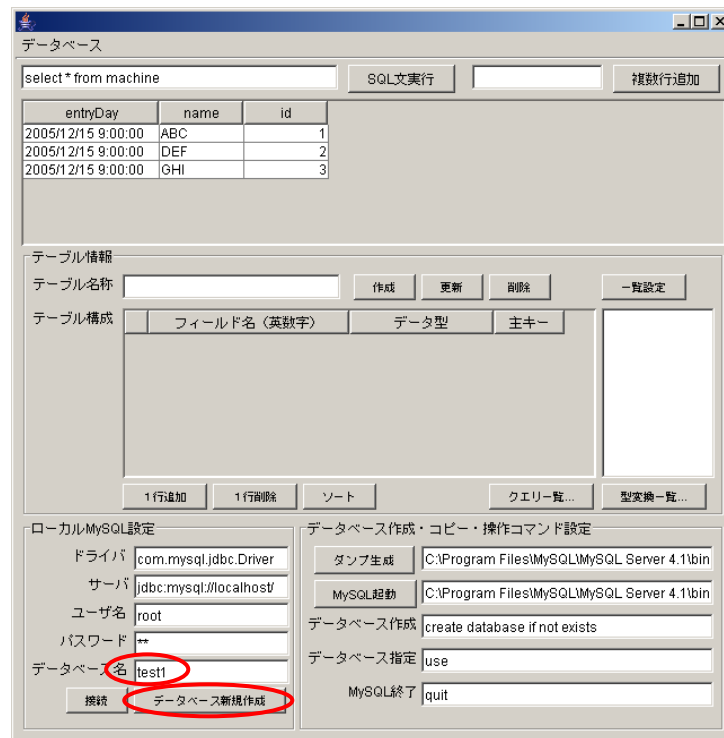


図 40 データベース test1 の新規作成

- ③ メニューから、[データベース] - [内容設定...] と選択し、前節で作成したバックアップファイル「backup.sql」を指定します。

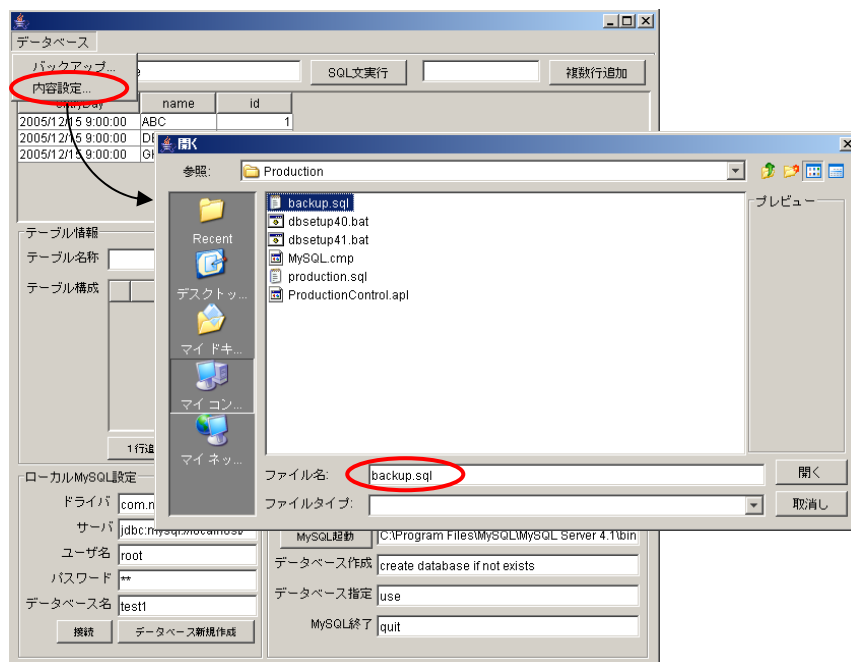


図 41 データベース内容設定

- ④ アプリケーションを終了します。

操 作

「test1」データベースが作成されていることを確認しましょう。

- ① **実行(設定可)**ボタンをクリックし、アプリケーションを起動します。
- ② 現在表示されているテーブルを削除します。
テーブル内で右クリック - [テーブル] - [列] - [全削除] とクリックします。
- ③ [接続]ボタンをクリックし、「test1」データベースに接続します。
- ④ [一覧設定]ボタンをクリックし、machine テーブルが作成されていることを確認します。
- ⑤ [SQL 文実行]ボタンをクリックして、テーブルの内容が設定されていることを確認します。

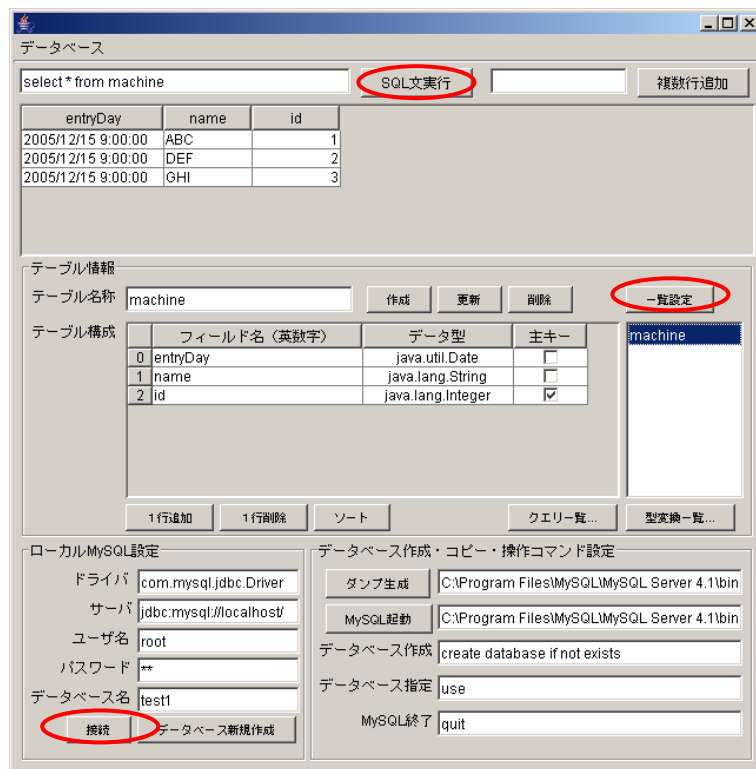


図 42 データベース内容設定結果の確認

- ⑥ アプリケーションを終了します。

2.9. データベーステーブル編集機能の追加

データベーステーブルは、UPDATE 文を実行することで編集することができます。

UPDATE <テーブル名> SET <カラム名> = <値>, ... WHERE 制約条件;

しかしながら、複数のレコードやフィールドを編集する場合には、エクセルのような表を書き換える方が容易です。「データベースアクセス」コンポーネントには、「検索結果を用いたテーブル更新処理」という以下の機能があります。

1. SELECT 文で取得したテーブルを内部に保持し、
2. そのテーブルを更新すると、自動的に MySQL データベーステーブルを更新する。

ここでは、この「検索結果を用いたテーブル更新処理」機能を利用し、アプリケーションにデータベーステーブル編集機能を追加することにします。下の図は、この機能の概略を示したものです。

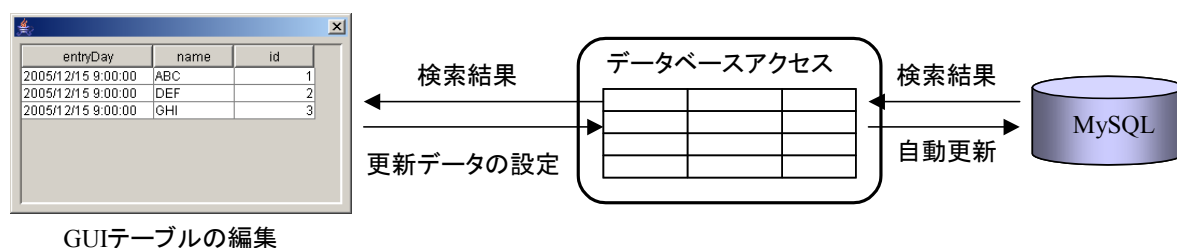


図 43 データベーステーブル編集機能の概略

SELECT 文で取得した検索結果を、「データベースアクセス」コンポーネント内部に保持するとともに、アプリケーションの GUI テーブルに設定します。ユーザの GUI テーブル編集結果に応じて、「データベースアクセス」コンポーネントが保持しているテーブルに値を設定します。すると、「データベースアクセス」コンポーネントによる MySQL データベーステーブルの更新が自動で行われます。

ここでは、データベースアクセスコンポーネントが持つ以下の3つのメソッドを使います。

メソッド	機能
startUpdate()	「検索結果を用いたテーブル更新処理」を開始します。
updateCell(Object, int, int)	「検索結果を用いたテーブル更新処理」の実行中に、「データベースアクセス」コンポーネントが保持しているテーブルのセル値を設定します。
finishUpdate()	「検索結果を用いたテーブル更新処理」を終了します。

そのために、まず、「ローカル MySQL 管理」複合コンポーネントへ移動し、これらを公開メソッドとして指定します。これらのメソッドは日本語化されていないので、メソッド選択ダイアログで「全メソッド対象」をチェックします。

操 作 メソッドを公開しましょう。

- ① 「ローカル MySQL 管理」複合コンポーネントへ移動します。
- ② 「ローカル MySQL 管理」複合コンポーネントを右クリック- [公開メソッド設定...] をクリックします。
- ③ 「データベースアクセス」コンポーネントを右クリック- [公開メソッド設定...] をクリックします。
- ④ [全メソッド対象] チェックボックスをチェックし、一覧から「finishUpdate()」「startUpdate()」「updateCell(Object, int, int)」を選択し追加(公開化)ボタンをクリックします。続けて設定ボタンをクリックします。

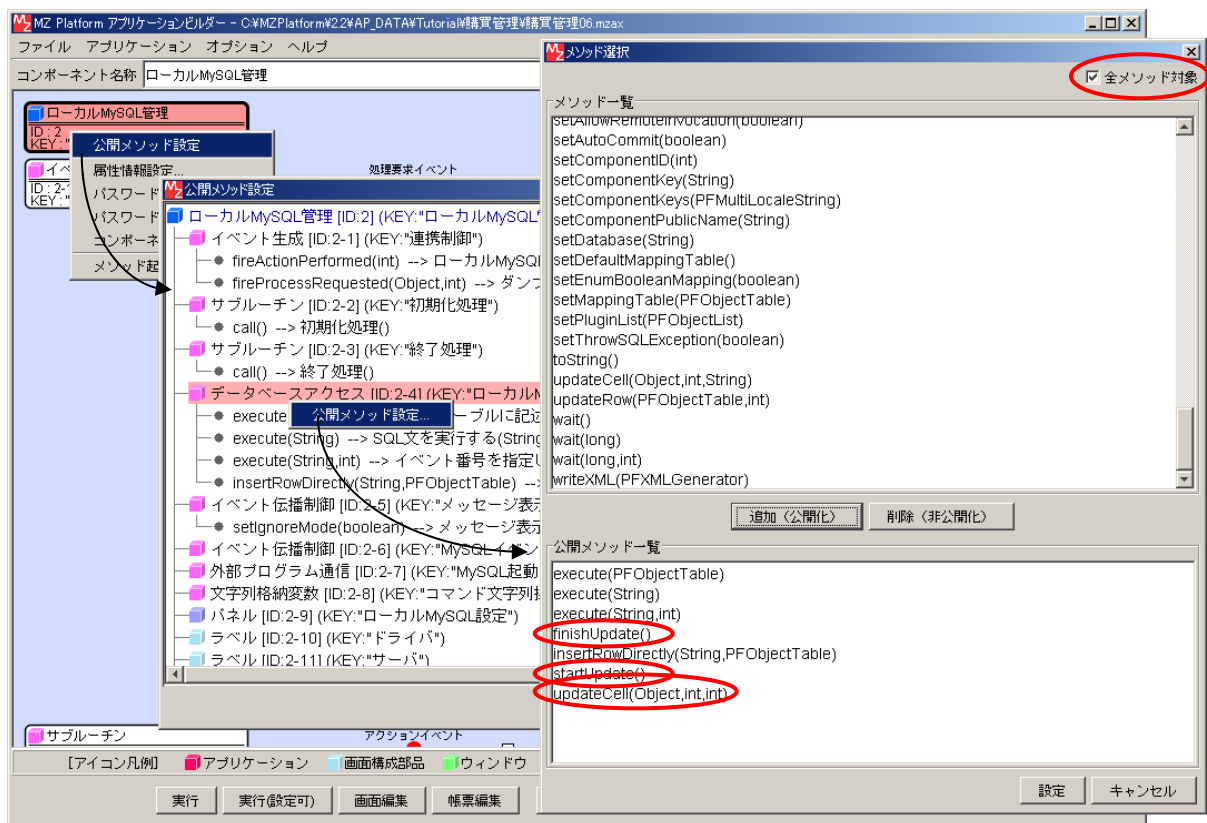


図 44 公開メソッドの設定

- ⑤ [公開メソッド設定] ダイアログを閉じ、アプリケーションビルダーのトップ画面に移動します。

それでは、データベーステーブル編集機能を追加します。⁷

追加するコンポーネント

- ダイアログ
- メニューアイテム
- テキストフィールド
- ボタン
- テーブル

追加する機能

1. テーブル編集メニューを選択したら、
2. データベースアクセスコンポーネントの「検索結果を用いたテーブル更新処理」を開始し、
3. ダイアログが表示される。
4. ダイアログ上のボタンをクリックするとテキストフィールドの SELECT 文を実行し、
5. 実行結果がダイアログ上の GUI テーブルに設定される。
6. GUI テーブルを編集すると、
7. データベースアクセスコンポーネントが内部に保持しているテーブルが更新される。
8. ダイアログを閉じると、
9. データベースアクセスコンポーネントの「検索結果を用いたテーブル更新処理」を終了し、
10. 編集していた GUI テーブルの内容をすべて削除する。

⁷ 購買管理 07.mzax

操 作

コンポーネントを追加し、接続しましょう。

① 必要なコンポーネントを追加します。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [ダイアログ] - [ダイアログ] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [メニュー] - [メニューアイテム] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [テキスト] - [テキストフィールド] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [ボタン] - [ボタン] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [テーブル] - [テーブル] とクリックします。

② 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。

[ダイアログ] コンポーネントの[title]を「テーブル編集」、

[メニューアイテム] コンポーネントの[text]を「テーブル編集...」、

[テキストフィールド] コンポーネントの[Columns]を「16」、[ComponentKey]を「SELECT 文」、

[ボタン] コンポーネントの[text]を「テーブル取得」、

[テーブル] コンポーネントの[ComponentKey]を「編集対象テーブル」

と設定します。

③ 下の表に示すようにコンポーネントを接続します。

テーブル編集メニューを選択したら検索結果を用いたテーブル更新処理を開始する ❶

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ メニューアイテム (ID:16, KEY:テーブル編集...)
発生イベント	アクションイベント
接続先コンポーネント ❶	■ ローカル MySQL 管理 (ID:2)
起動メソッド	startUpdate()

ダイアログを表示する ❷

接続先コンポーネント ❷	■ ダイアログ (ID:15, KEY:テーブル編集)
起動メソッド	ダイアログを表示する()

ボタンをクリックするとテキストフィールドに入力された SELECT 文を実行する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ボタン (ID:18, KEY:テーブル取得)
発生イベント	アクションイベント
接続先コンポーネント	■ ローカル MySQL 管理 (ID:2)
起動メソッド	イベント番号を指定して SQL 文を実行する (String, int)
<引数 0>	取得方法 : メソッド戻り値 コンポーネント : テキストフィールド (ID:17, KEY:SELECT 文) メソッド/値 : テキストを取得する
<引数 1>	取得方法 : 固定値 メソッド/値 : 2

実行結果をテーブルに設定する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ローカル MySQL 管理 (ID:2)
発生イベント	データ生成イベント
接続先コンポーネント	■ テーブル (ID:19, KEY:編集対象テーブル)
起動メソッド	テーブルデータを設定する (PFObjectTable)
<引数>	説明 : テーブルデータ 取得方法 : イベント内包 メソッド/値 : イベント対象データ
イベント番号	2

SELECT 文 (SQL 文) を実行するときのイベント番号を 2 としましたので、「ローカル MySQL 管理」複合コンポーネントから発生するデータ生成イベントの番号も 2 を指定しています。

テーブルを編集すると「データベースアクセス」コンポーネント内部のテーブルが更新される

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ テーブル (ID:19, KEY:編集対象テーブル)
発生イベント	データ更新イベント
接続先コンポーネント	■ ローカル MySQL 管理 (ID:2)
起動メソッド	updateCell(Object, int, int)
<引数 0>	取得方法: イベント内包 メソッド/値: 更新データ
<引数 1>	取得方法: メソッド戻り値 コンポーネント: テーブル (ID:19, KEY:編集対象テーブル) メソッド/値: 選択行の位置を取得する
<引数 2>	取得方法: メソッド戻り値 コンポーネント: テーブル (ID:19, KEY:編集対象テーブル) メソッド/値: 選択列の位置を取得する
イベント番号	0

ダイアログを閉じると検索結果を用いたテーブル更新処理を終了する ①

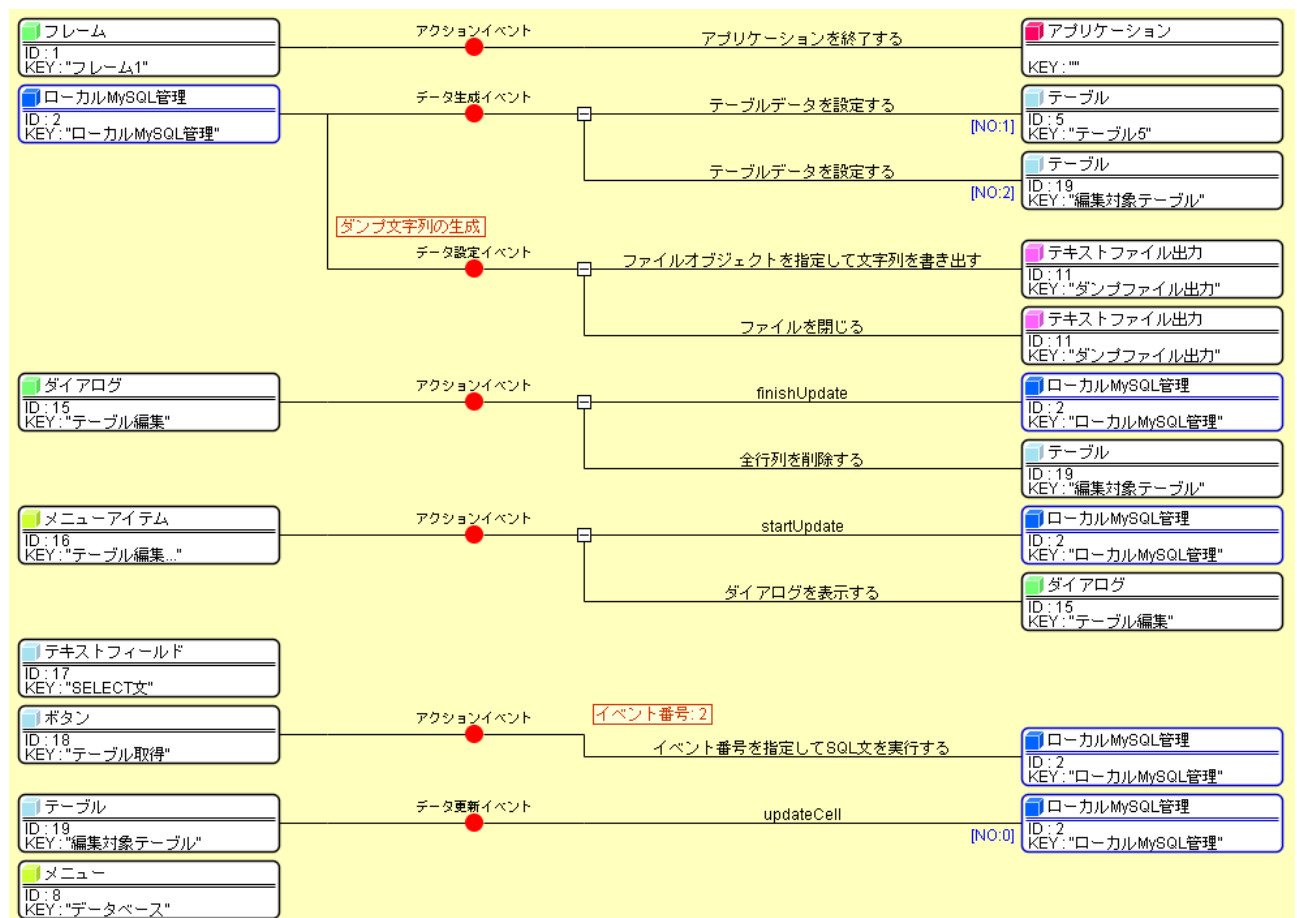
接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ダイアログ (ID:15, KEY:テーブル編集)
発生イベント	アクションイベント
接続先コンポーネント ①	■ ローカル MySQL 管理 (ID:2)
起動メソッド	finishUpdate()

GUI テーブルの内容を全て削除する ②

接続先コンポーネント ②	■ テーブル (ID:19, KEY:編集対象テーブル)
起動メソッド	全行列を削除する()

確認

接続関係と配置は以下のようになります。



操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 「メニューアイテム」コンポーネントをメニューに追加します。左側のコンポーネント一覧から「メニュー」コンポーネントを右クリック－[コンポーネント追加]－[メニューアイテム]とクリックします。
- ③ 「ダイアログ」コンポーネントをフレームに追加します。左側のコンポーネント一覧から「フレーム」コンポーネントを右クリック－[コンポーネント追加]－[ダイアログ]とクリックします。
- ④ 左側のコンポーネント一覧から「ダイアログ」コンポーネントを右クリック－[コンポーネント追加]－[テキストフィールド]とクリックします。
- ⑤ 左側のコンポーネント一覧から「ダイアログ」コンポーネントを右クリック－[コンポーネント追加]－[ボタン]とクリックします。
- ⑥ 左側のコンポーネント一覧から「ダイアログ」コンポーネントを右クリック－[コンポーネント追加]－[テーブル]とクリックします。
- ⑦ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

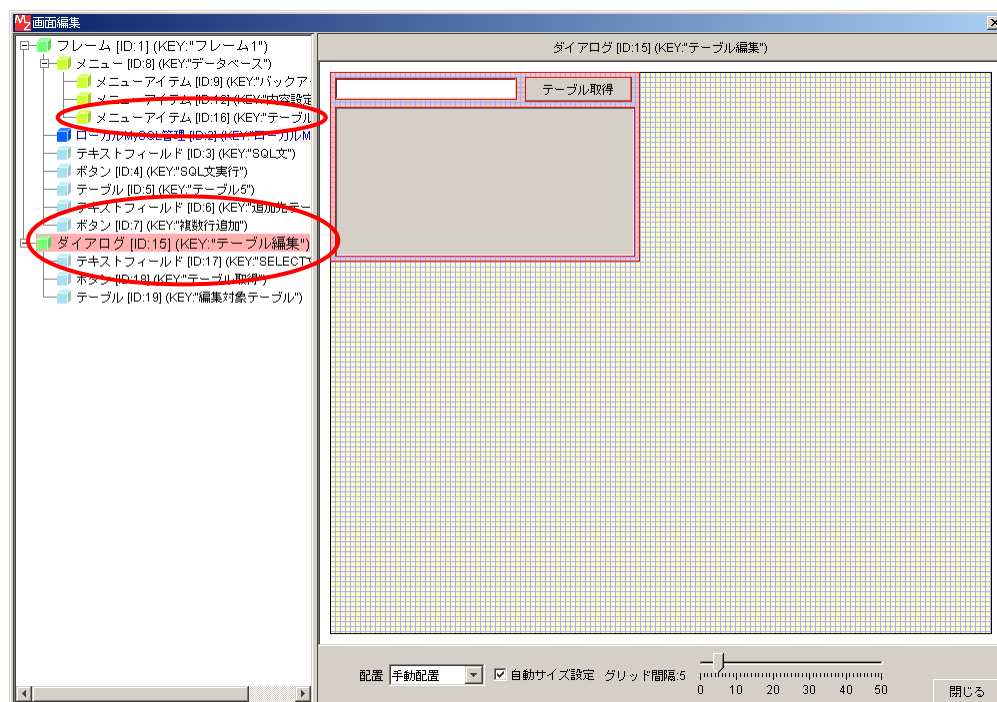


図 46 データベーステーブル編集機能の追加(ダイアログの画面配置とメニューへの追加)

操 作

アプリケーションの動作を確認しましょう。

- ① [実行]もしくは[実行(設定可)]ボタンをクリックし、アプリケーションを起動します。
- ② [データベース名]に「test」と入力し、[接続]ボタンをクリックします。
- ③ メニューから[データベース] - [テーブル編集...]を選択し、[テーブル編集]ダイアログを表示させます。

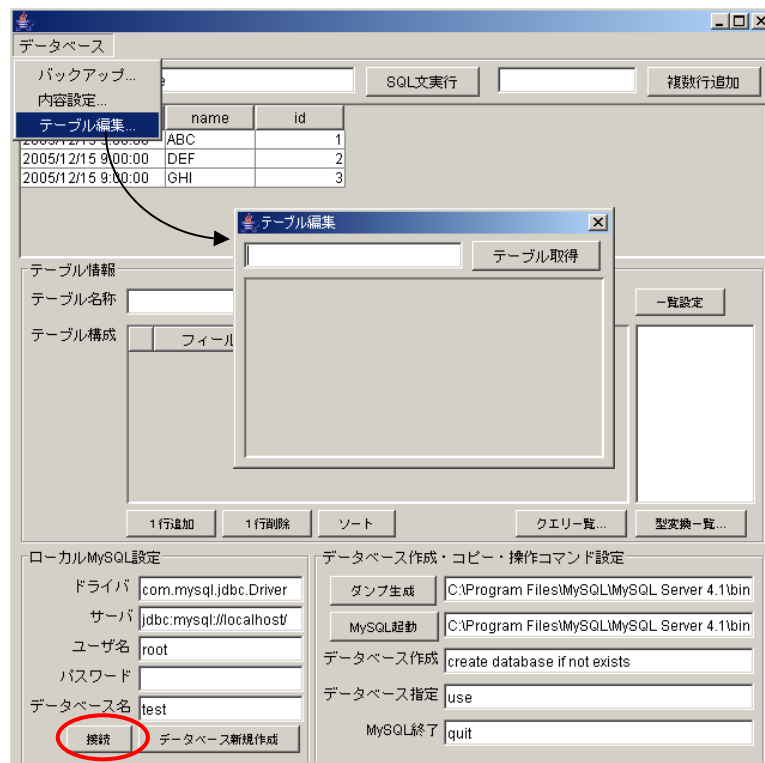


図 47 テーブル編集ダイアログの表示

- ④ [テーブル編集] ダイアログのテキストフィールドに「select * from machine」と入力し、[テーブル取得]ボタンをクリックします。

!!! 注意事項 !!!

取得したテーブルには、主キーのフィールドが含まれていなくてはなりません。

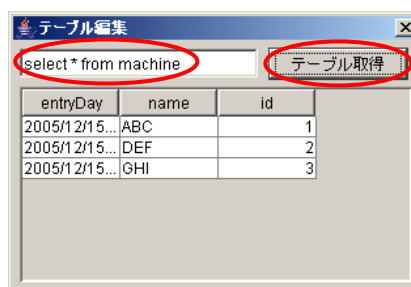


図 48 テーブルの取得

- ⑤ テーブルを適当に編集し、ダイアログを閉じます。

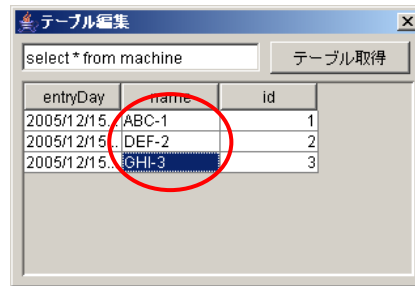


図 49 テーブルの編集

- ⑥ [SQL 文実行] ボタンをクリックして検索を実行し、データベーステーブルの内容が更新されていることを確認します。

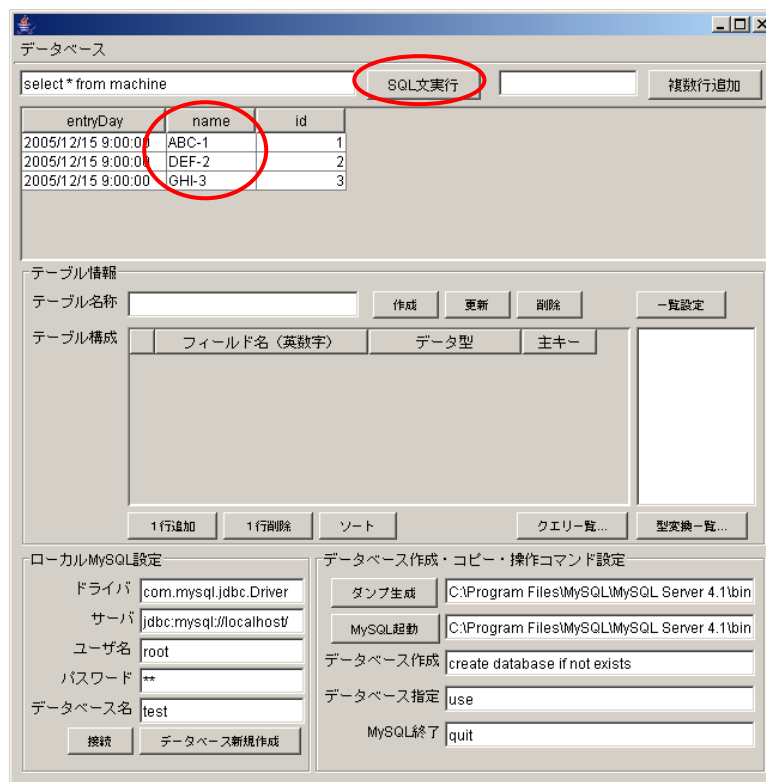


図 50 データベーステーブル更新結果の確認

- ⑦ 確認後、アプリケーションを終了します。

3. 発注依頼部の作成

データベース連携部に続いて、発注依頼部を作成します。

3.1. 購買管理データベースの作成

はじめに、購買管理データベースを作成します。

操 作 kobai データベースを作成しましょう。

- ① **実行**もしくは**実行(設定可)**ボタンをクリックし、アプリケーションを起動します。
- ② [データベース名]に「kobai」と入力し、[データベース新規作成]ボタンをクリックします。
- ③ メニューから[データベース] - [内容設定...]を選択します。Javaの論理型(java.lang.Boolean)のデータをenum('true','false')に変換するように設定した場合(14 ページ参照)には、AP_DATA¥Tutorial¥購買管理フォルダ内の「kobai-enum.sql」ファイルを、BOOLEANとして扱うように設定した場合には「kobai.sql」を指定します。
- ④ [接続]ボタンをクリックし「kobai」データベースに接続します。
- ⑤ [一覧設定]ボタンをクリックし、以下のデータベーステーブルが作成されていることを確認します。

- irai(発注依頼)

項目	主キー	注文番号	注文日	発注部門コード	購買品コード	メーカー名
フィールド名	id	kobaiCD	iraibi	bumonCD	hinmokuCD	maker
データ型	Integer	String	Date	String	String	String

規格仕様	希望納期	発注数量	単価	金額
kikaku	kiboNouki	kiboSuryo	kiboTanka	kiboKingaku
String	Date	Integer	Double	Double

(仮)注文先コード	製品設備コード	検査の有無	部門承認
kiboKobaisakiCD	seihinCD	kiboKensa	shonin
String	String	Boolean	Boolean

- kobai(購買管理)

項目	主キー	注文番号	注文先コード	検査の有無	納期回答	発注数量
フィールド名	id	kobaiCD	kobaisakiCD	kensa	nouki	suryo
データ型	Integer	String	String	Boolean	Date	Integer

発注単価	発注合計	納入日	受入担当者	完了日
tanka	kingaku	nonyubi	ukeireTanto	nouki
Double	Double	Date	String	Date

- keihi(経費管理)

項目	主キー	注文番号	経費部門コード	費目コード
フィールド名	id	kobaiCD	keihiBumonCD	himokuCD
データ型	Integer	String	String	String

- kobai_mstr(購買マスター)

項目	主キー	購買品コード	中区分コード	品名	メーカー名	規格仕様
フィールド名	id	hinmokuCD	chuKubunCD	hinmei	maker	kikaku
データ型	Integer	String	String	String	String	String

単価
tanka
Double

- kubun_mstr(区分マスター)

項目	主キー	大区分コード	大区分	中区分コード	中区分
フィールド名	id	daiKubunCD	daiKubun	chuKubunCD	chuKubun
データ型	Integer	String	String	String	String

区分 ID
kubunID
String

- kobaisaki_mstr(購買先マスター)

項目	主キー	注文先コード	注文先	指定納品書
フィールド名	id	kobaisakiCD	kobaisaki	yoshiki
データ型	Integer	String	String	String

- himoku_mstr(費目マスター)

項目	主キー	費目コード	費目
フィールド名	id	himokuCD	himoku
データ型	Integer	String	String

- seihin_mstr(製品設備マスター)

項目	主キー	製品設備コード	製品設備名
フィールド名	id	seihinCD	seihin
データ型	Integer	String	String

- bumon_mstr(部門マスター)

項目	主キー	部門コード	部門名	承認者
フィールド名	id	bumonCD	bumon	shoninID
データ型	Integer	String	String	String

これらのテーブルのフィールドのうち、参照関係にあるものは同じ色で示してあります。

- ⑥ 確認後、アプリケーションを終了します。

3.2. 「発注依頼」複合コンポーネントの作成

発注依頼部は、複合コンポーネントとして作成することになります。

操 作

「発注依頼」複合コンポーネントを作成しましょう。

- ① 複合コンポーネントを追加します。
作業領域で右クリック - [複合コンポーネント追加] - [コンポーネント生成] とクリックします。
- ② 作成した複合コンポーネントをダブルクリックし、複合コンポーネントの中へ移動します。
- ③ コンポーネント名称を設定します。
ビルダー画面上部のコンポーネント名称欄に「発注依頼」と入力し【Enter】キーを押します。
- ④ コンポーネントキーを設定します。
属性情報設定画面を表示し、[ComponentKey]を「発注依頼」と設定します。

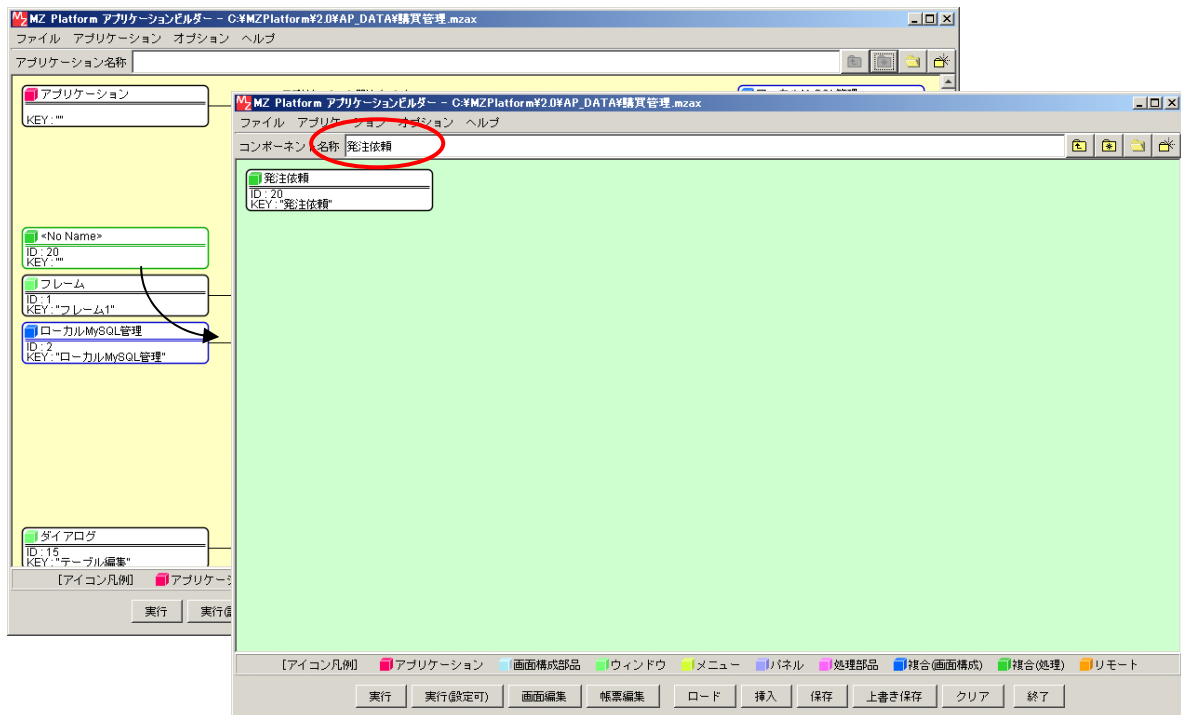


図 51 複合コンポーネントの生成

3.3. 初期コンポーネントの追加

この複合コンポーネントは、トップ階層にある「ローカル MySQL 管理」複合コンポーネントとの間で、イベントの送受信によるデータの交換を行います。このように、上位階層とイベントの送受信を行うことが想定される複合コンポーネントの場合、最初に以下のコンポーネントを追加しておく便利です。

追加するコンポーネント

- フレーム
- イベント生成(受信用)
- イベント生成(送信用)
- サブルーチン(初期化处理用)
- サブルーチン(終了処理用)

操 作

コンポーネントを追加し、コンポーネントキーを設定しましょう。

① 必要なコンポーネントを追加します。

作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [ウィンドウ] - [フレーム] とクリックします。

作業領域で右クリック - [コンポーネント追加] - [処理部品] - [イベント] - [イベント生成] とクリックします。これを受信用とします。

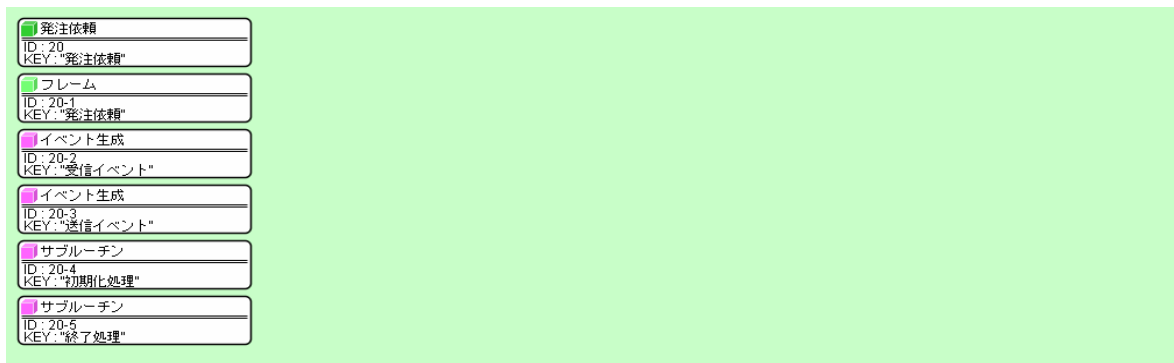
作業領域で右クリック - [コンポーネント追加] - [処理部品] - [イベント] - [イベント生成] とクリックします。これを送信用とします。

作業領域で右クリック - [コンポーネント追加] - [処理部品] - [サブルーチン] - [サブルーチン] とクリックします。これを初期化处理用とします。

作業領域で右クリック - [コンポーネント追加] - [処理部品] - [サブルーチン] - [サブルーチン] とクリックします。これを終了処理用とします。

- ② 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。

[フレーム] コンポーネントの[title]を「発注依頼」、[ComponentKey]を「発注依頼」、
受信用として追加した [イベント生成] コンポーネントの[ComponentKey]を「受信イベント」、
送信用として追加した [イベント生成] コンポーネントの[ComponentKey]を「送信イベント」、
初期化处理用として追加した [サブルーチン] コンポーネントの[ComponentKey]を「初期化处理」、
終了処理用として追加した [サブルーチン] コンポーネントの[ComponentKey]を「終了処理」
と設定します。



■ 発注依頼
ID: 20
KEY: "発注依頼"
■ フレーム
ID: 20-1
KEY: "発注依頼"
■ イベント生成
ID: 20-2
KEY: "受信イベント"
■ イベント生成
ID: 20-3
KEY: "送信イベント"
■ サブルーチン
ID: 20-4
KEY: "初期化处理"
■ サブルーチン
ID: 20-5
KEY: "終了処理"

図 52 初期コンポーネントの追加

3.4. 公開メソッドおよびコンポーネント接続の初期設定

トップ階層から起動されることが想定されるメソッドを、最初に公開メソッドとして設定しておきます。

コンポーネント	公開するメソッド	目的
フレーム	フレームを表示する() フレームを閉じる()	
イベント生成 (受信イベント)	データ生成イベントの発生 (Object, int)	トップ階層の「ローカル MySQL 管理」複合コンポーネントから発生するデータ生成イベントを伝達するため
サブルーチン (初期化处理)	処理を呼び出す()	トップ階層から初期化处理を起動するため
サブルーチン(終了処理)	処理を呼び出す()	トップ階層から終了処理を起動するため

操 作 メソッドを公開しましょう。

- ① 「発注依頼」複合コンポーネントを右クリック- [公開メソッド設定...] をクリックします。
- ② 「フレーム」コンポーネントを右クリック- [公開メソッド設定...] をクリックします。
- ③ 一覧から「フレームを表示する()」「フレームを閉じる()」を選択し **追加(公開化)** ボタンをクリックします。続けて **設定** ボタンをクリックします。
- ④ 「イベント生成(受信イベント)」コンポーネントを右クリック- [公開メソッド設定...] をクリックします。
- ⑤ 一覧から「データ生成イベントの発生(Object, int)」を選択し **追加(公開化)** ボタンをクリックします。続けて **設定** ボタンをクリックします。
- ⑥ 「サブルーチン(初期化处理)」コンポーネントを右クリック- [公開メソッド設定...] をクリックします。
- ⑦ 一覧から「処理を呼び出す()」を選択し **追加(公開化)** ボタンをクリックします。続けて **設定** ボタンをクリックします。
- ⑧ 終了処理と区別するため、公開メソッド名を変更します。
「call()→処理を呼び出す()」を右クリック- [公開メソッド名変更...] をクリックし、「初期化处理を呼び出す」と入力します。

- ⑨ 「サブルーチン(終了処理)」コンポーネントを右クリック- [公開メソッド設定...] をクリックします。
- ⑩ 一覧から「処理を呼び出す()」を選択し追加(公開化)ボタンをクリックします。
続けて設定ボタンをクリックします。
- ⑪ 初期化処理と区別するため、公開メソッド名を変更します。
「call() --> 処理を呼び出す()」を右クリック- [公開メソッド名変更...] をクリックし、
「終了処理を呼び出す」と入力します。
- ⑫ [公開メソッド設定] ダイアログを閉じます。

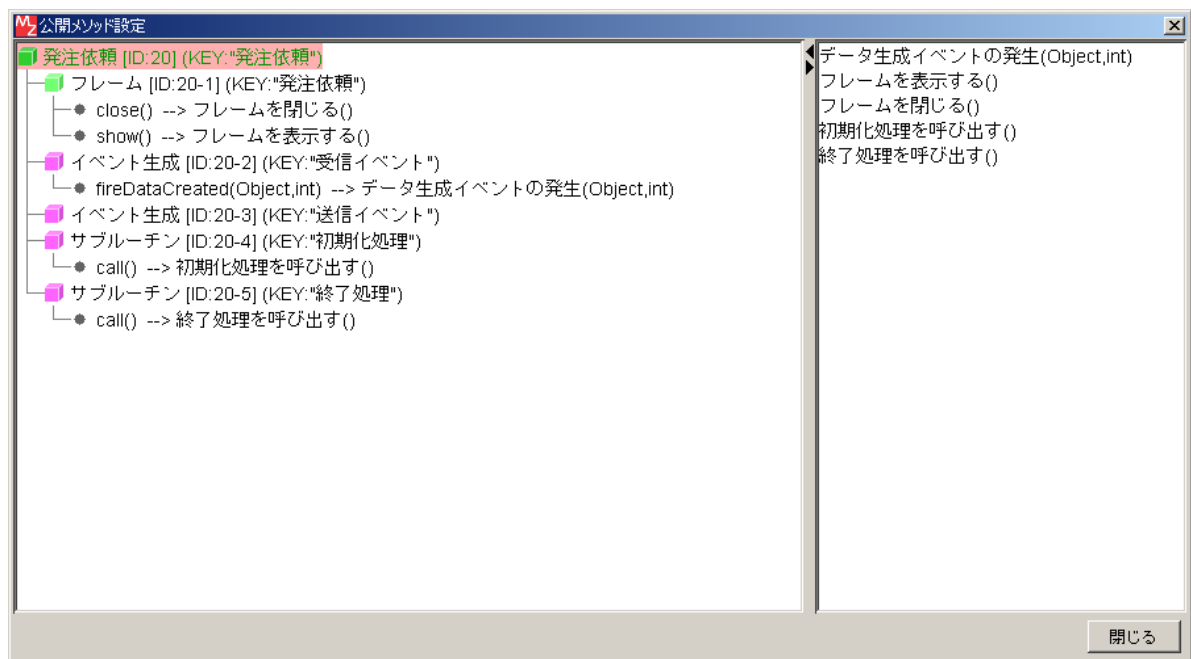


図 53 公開メソッドの設定コンポーネント接続の初期設定

また、以下のコンポーネントから発生するイベントの接続先を設定しておきます。⁸

- イベント生成(送信イベント)

トップ階層の「ローカル MySQL 管理」複合コンポーネントが持つ「イベント番号を指定して SQL 文を実行する(String, int)」メソッドへの引数を伝達するため、データ生成イベントを上位階層へ伝播させるようにしておきます。ここで使用するイベントは、必ずしもデータ生成イベントでなくても構いません。

- サブルーチン(終了処理)

終了処理としてフレームを閉じるようにしておきます。

操 作

コンポーネントを接続しましょう。

① 下の表に示すようにコンポーネントを接続します。

データ生成イベントを上位階層へ伝播させる

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ イベント生成 (ID:20-3, KEY:送信イベント)
発生イベント	データ生成イベント
接続先コンポーネント	■ 発注依頼 (ID:20, KEY:発注依頼)
起動メソッド	イベントを伝播させる(PFEvent)
<引数>	説明: 対象イベント 取得方法: イベント

フレームを閉じる

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ サブルーチン (ID:20-5, KEY:終了処理)
発生イベント	アクションイベント
接続先コンポーネント	■ フレーム (ID:20-1, KEY:発注依頼)
起動メソッド	フレームを閉じる()

⁸ 購買管理 08.mzax

確認

接続関係と配置は以下のようになります。

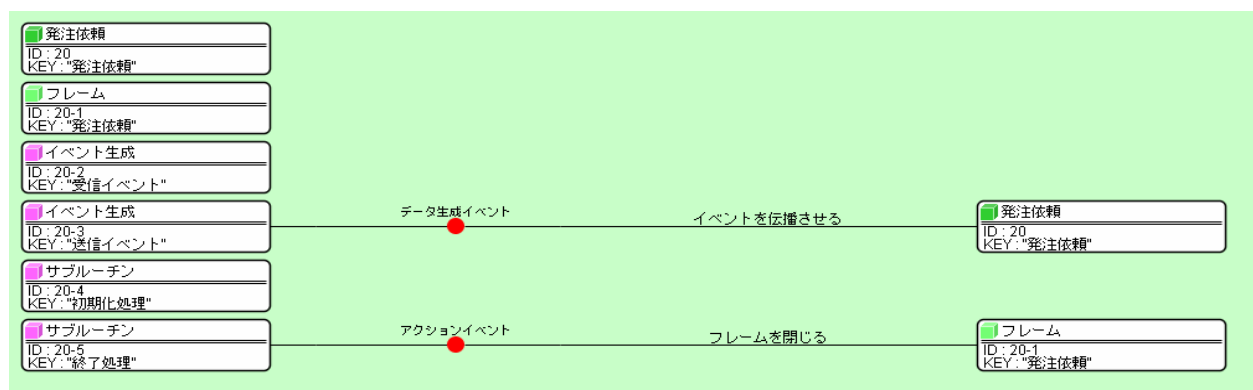


図 54 コンポーネント接続の初期設定

3.5. トップ階層におけるイベント接続先の初期設定

アプリケーションビルダーのトップ階層へ移動し必要となるイベントの接続先を設定しておきます。⁹

操 作 コンポーネントを接続しましょう。

- ① アプリケーションビルダーのトップ画面に移動します。
- ② 下の表に示すようにコンポーネントを接続します。

初期化処理を呼び出す

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ アプリケーション
発生イベント	アプリケーション開始イベント
接続先コンポーネント	■ 発注依頼 (ID:20, KEY:発注依頼)
起動メソッド	初期化処理を呼び出す()

終了処理を呼び出す

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ アプリケーション
発生イベント	アプリケーション終了イベント
接続先コンポーネント	■ 発注依頼 (ID:20, KEY:発注依頼)
起動メソッド	終了処理を呼び出す()

⁹ 購買管理 09.mzax

イベント番号を指定して SQL 文を実行する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ 発注依頼 (ID:20, KEY:発注依頼)
発生イベント	データ生成イベント
接続先コンポーネント	■ ローカル MySQL 管理 (ID:2)
起動メソッド	イベント番号を指定して SQL 文を実行する (String, int)
<引数 0>	取得方法: イベント内包 メソッド/値: イベント対象データ
<引数 1>	取得方法: イベント内包 メソッド/値: イベント番号

データ生成イベントの発生を起動する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ローカル MySQL 管理 (ID:2)
発生イベント	データ生成イベント
接続先コンポーネント	■ 発注依頼 (ID:20, KEY:発注依頼)
起動メソッド	データ生成イベントの発生 (Object, int)
<引数 0>	説明: 生成データ 取得方法: イベント内包 メソッド/値: イベント対象データ
<引数 1>	説明: イベント番号 取得方法: イベント内包 メソッド/値: イベント番号

確認

接続関係と配置は以下のようになります。

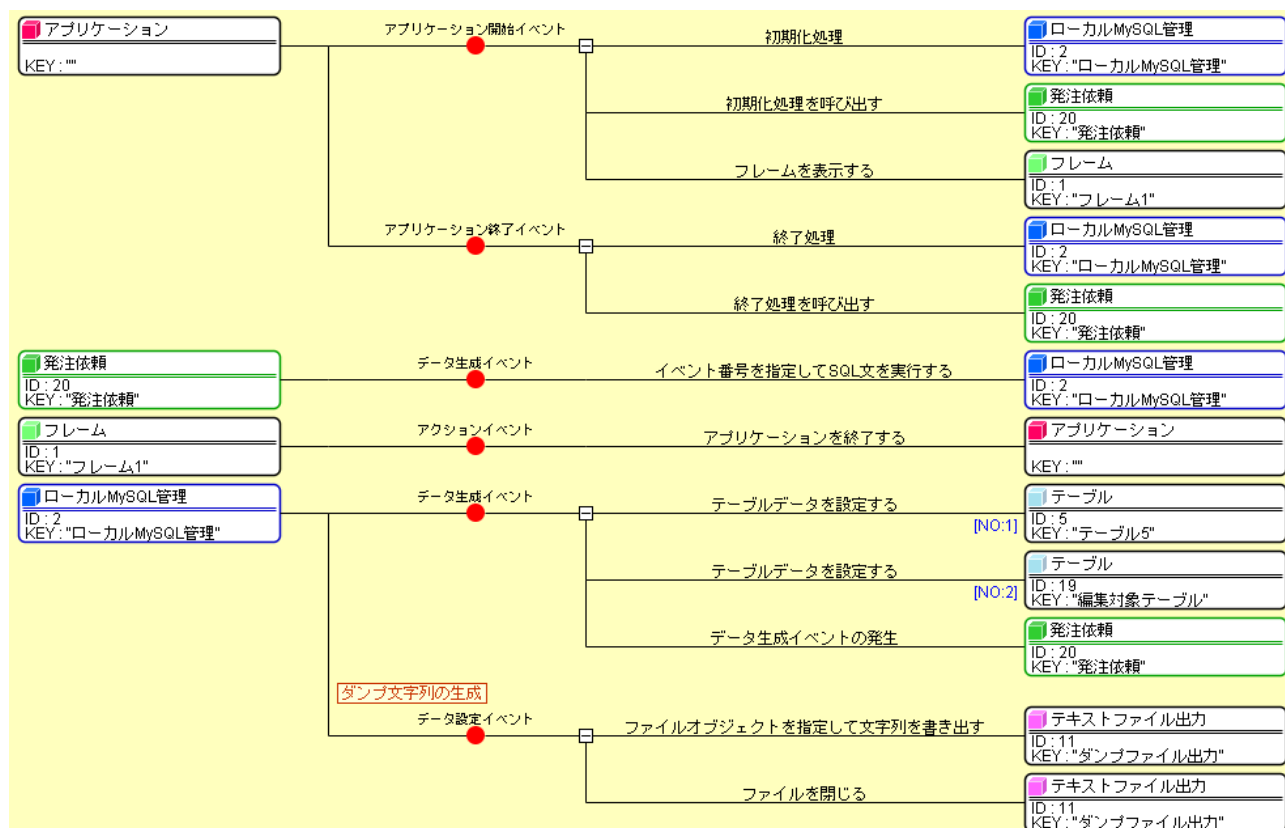


図 55 トップ階層におけるイベントの接続先と起動メソッド

3.6. 発注依頼フレームのメニュー登録

発注依頼フレームをメニューから起動できるようにします。¹⁰

追加するコンポーネント

- メニュー
- メニューアイテム

追加する機能

1. 発注依頼メニューを選択したら、
2. 発注依頼フレームが表示される。

操 作 コンポーネントを追加し、接続しましょう。

- ① 必要なコンポーネントを追加します。
作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [メニュー] - [メニュー]
とクリックします。
作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [メニュー] -
[メニューアイテム] とクリックします。
- ② 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。
[メニュー] コンポーネントの[text]を「購買管理」、
[メニューアイテム] コンポーネントの[text]を「発注依頼...」と設定します。
- ③ 下の表に示すようにコンポーネントを接続します。

発注依頼メニューを選択したら発注依頼フレームを表示する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■メニューアイテム (ID:22, KEY:発注依頼...)
発生イベント	アクションイベント
接続先コンポーネント	■発注依頼 (ID:20, KEY:発注依頼)
起動メソッド	フレームを表示する()

¹⁰ 購買管理 10.mzax

確認

接続関係と配置は以下のようになります。

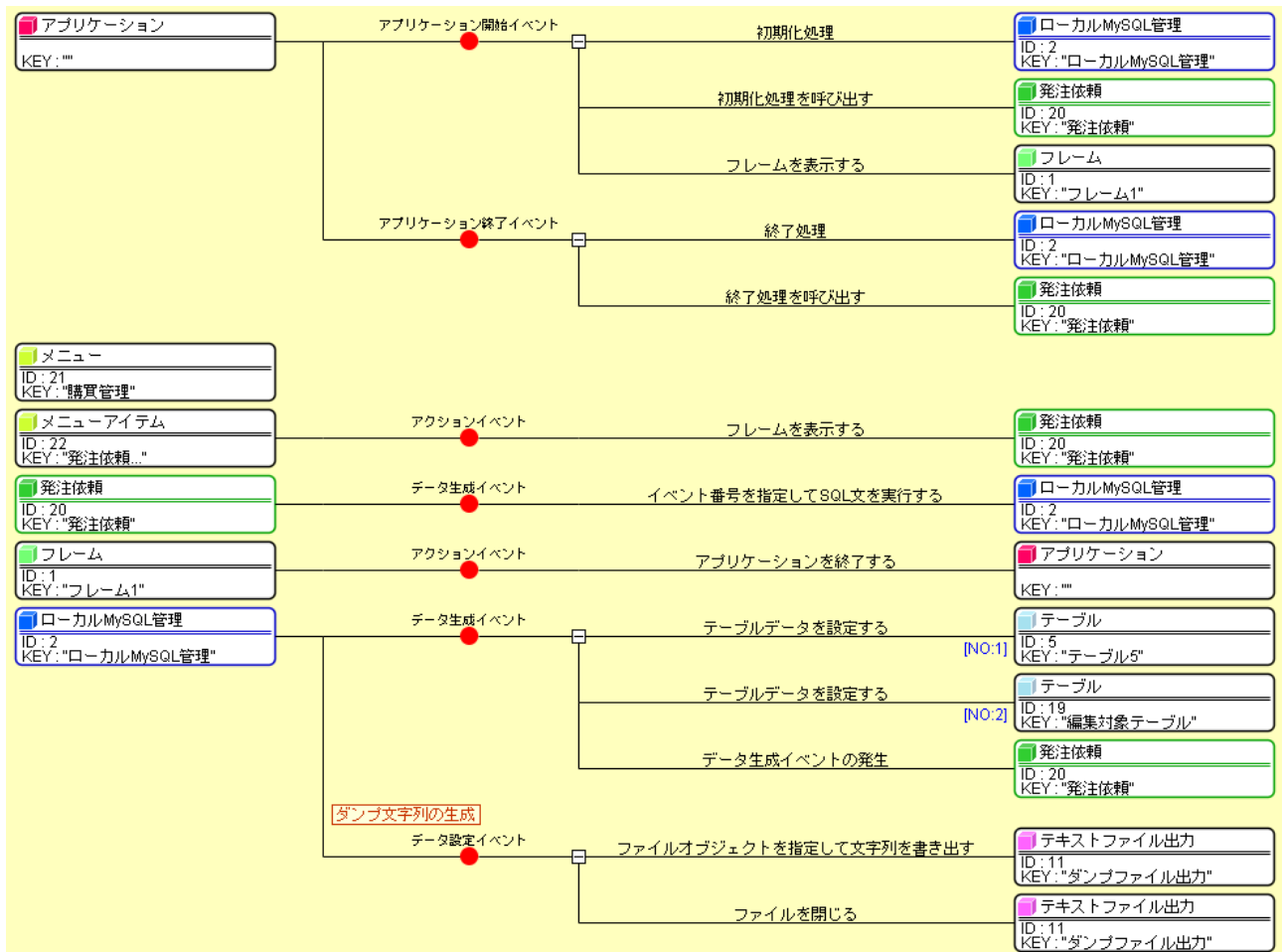


図 56 発注依頼フレームのメニュー登録(コンポーネント接続)

操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 「メニュー」コンポーネントをフレームに追加します。左側のコンポーネント一覧から「フレーム」コンポーネントを右クリック → [コンポーネント追加] → [メニュー] とクリックします。
- ③ 「メニューアイテム」コンポーネントをメニューに追加します。左側のコンポーネント一覧から「メニュー」コンポーネントを右クリック → [コンポーネント追加] → [メニューアイテム] とクリックします。
- ④ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

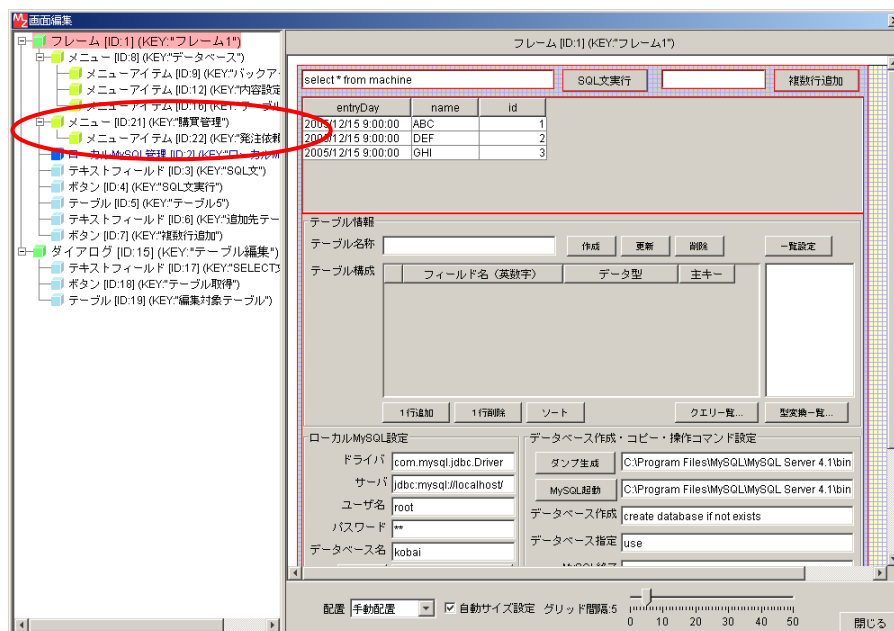


図 57 発注依頼フレームのメニュー登録(メニューの追加)

操 作

アプリケーションの動作を確認しましょう。

- ① **実行**もしくは**実行(設定可)** ボタンをクリックし、アプリケーションを起動します。
- ② メニューから [購買管理] → [発注依頼...] を選択し、[発注依頼] フレームが表示されることを確認します。
- ③ メインウィンドウを終了し [発注依頼] フレームが閉じることを確認します。

3.7. 参照機能の作成と登録機能作成の準備

発注依頼部では、kobai データベースへアクセスし、irai テーブルの検索、レコードの登録、削除、更新、そして発注依頼の伝票発行を行います。その概観は以下の通りです。

注文番号	注文日	発注部門コード	購買品コード	発注数量	単価	金額	部門承認
A1	2005/12/20	S1	P1	5	10,000	50,000	☐

図 58 発注依頼部概観

SQL 文を作成するときには、あらかじめ定型文を用意しておき、必要に応じて文字列の代入や挿入を行うようにしておくとう便利です。発注依頼部のテーブルでは、kobaiCD(注文番号)、iraibi(注文日)、bumonCD(発注部門コード)、hinmokuCD(購買品コード)、kiboSuryo(発注数量)、kiboTanka(単価)、kiboKingaku(金額)、shonin(部門承認)のフィールドを使います。したがって、注文番号を指定してレコードを検索するには以下の定型文を用意し、必要に応じて「KOBACD」の部分を入れ替えます。

```
select kobaiCD, iraibi, bumonCD, hinmokuCD, kiboSuryo, kiboTanka, kiboKingaku, shonin from irai
where kobaiCD='KOBACD'
```

まず、参照機能の作成と登録機能作成の準備を行います。¹¹

追加するコンポーネント

- ラベル
- ボタン(参照用)
- ボタン(登録用)
- テーブル(画面表示用)
- テーブル(登録用)
- 文字列格納変数(SQL 定型文を設定し、「KOBACD」を検索対象の値で置き換えるために使う)
- テーブル格納変数
- サブルーチン(検索結果をテーブルに設定する処理を記述する)

追加する機能

1. [参照]ボタンをクリックすると、
2. 文字列格納変数コンポーネントに SQL 定型文を設定し、
3. 「KOBACD」を画面表示用テーブルの 0 行 0 列のセル値で置き換え、
4. その SQL 文をイベント番号 3 で実行し、
5. 取得した検索結果を、画面表示用と登録用の 2 つのテーブルに設定する。

¹¹ 購買管理 11.mzax

操 作

コンポーネントを追加し、接続しましょう。

① 「発注依頼」複合コンポーネントへ移動します。

② 必要なコンポーネントを追加します。

作業領域で右クリック－[コンポーネント追加]－[画面構成部品]－[テキスト]－[ラベル]
とクリックします。

作業領域で右クリック－[コンポーネント追加]－[画面構成部品]－[ボタン]－[ボタン]
とクリックします。これを参照用とします。

作業領域で右クリック－[コンポーネント追加]－[画面構成部品]－[ボタン]－[ボタン]
とクリックします。これを登録用とします。

作業領域で右クリック－[コンポーネント追加]－[画面構成部品]－[テーブル]－[テーブル]
とクリックします。これを画面表示用とします。

作業領域で右クリック－[コンポーネント追加]－[画面構成部品]－[テーブル]－[テーブル]
とクリックします。これを登録用とします。

作業領域で右クリック－[コンポーネント追加]－[処理部品]－[変数]－[文字列格納変数]
とクリックします。

作業領域で右クリック－[コンポーネント追加]－[処理部品]－[変数]－[テーブル格納変数]
とクリックします。

作業領域で右クリック－[コンポーネント追加]－[処理部品]－[サブルーチン]－[サブルーチン]
とクリックします。

③ 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。

[ラベル] コンポーネントの[text]を「select kobaiCD, iraibi, bumonCD, hinmokuCD, kiboSuryo,
kiboTanka, kiboKingaku, shonin from irai where kobaiCD = 'KOBACD'」、

[ComponentKey]を「SQL 定型文」、

参照用として追加した [ボタン] コンポーネントの[text]を「参照」、

登録用として追加した [ボタン] コンポーネントの[text]を「登録」、

画面表示用として追加した [テーブル] コンポーネントの[ComponentKey]を「画面表示用」、

登録用として追加した [テーブル] コンポーネントの[ComponentKey]を「登録用」、

[サブルーチン]コンポーネントの[ComponentKey]を「参照データ取得」

と設定します。

- ④ 下の表に示すようにコンポーネントを接続します。

参照ボタンをクリックすると文字列格納変数に SELECT SQL 定型文を設定する ❶

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ボタン (ID:20-8, KEY:参照)
発生イベント	アクションイベント
接続先コンポーネント ❶	■ 文字列格納変数 (ID:20-12)
起動メソッド	文字列を設定する(イベント発生なし) (String)
<引数>	取得方法: メソッド戻り値 コンポーネント: ラベル (ID:20-7, KEY:SQL 定型文) メソッド/値: ラベルのテキスト文字列を取得する

画面表示用テーブルの 0 行 0 列の値を返す ❷

接続先コンポーネント ❷	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	指定セルの値を取得する (int, int)
<引数 0>	取得方法: 固定値 メソッド/値: 0
<引数 1>	取得方法: 固定値 メソッド/値: 0

KOBAICD を画面表示用テーブルの 0 行 0 列の値に置き換える ❸

接続先コンポーネント ❸	■ 文字列格納変数 (ID:20-12)
起動メソッド	正規表現に適合する最初の文字列を置換する (String, String)
<引数 0>	取得方法: 固定値 メソッド/値: KOBAICD
<引数 1>	取得方法: メソッド処理結果 メソッド/値: 指定セルの値を取得する

SQL 文をイベント番号 3 で実行する ④

接続先コンポーネント ④	■ イベント生成 (ID:20-3, KEY:送信イベント)
起動メソッド	データ生成イベントの発生 (Object, int)
<引数 0>	説明：生成データ 取得方法：メソッド戻り値 コンポーネント：文字列格納変数 (ID:20-12) メソッド/値：文字列を取得する
<引数 1>	説明：イベント番号 取得方法：固定値 メソッド/値 3

イベントを参照データ取得サブルーチンに渡して処理を呼び出す

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ イベント生成 (ID:20-2, KEY:受信イベント)
発生イベント	データ生成イベント
接続先コンポーネント	■ サブルーチン (ID:20-14, KEY:参照データ取得)
起動メソッド	イベントを渡して処理を呼び出す (PFEvent)
<引数>	説明：イベント 取得方法：イベント
イベント番号	3

取得した検索結果を画面表示用テーブルに設定する ①

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ サブルーチン (ID:20-14, KEY:参照データ取得)
発生イベント	データ生成イベント
接続先コンポーネント ①	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	テーブルデータを設定する (PFObjectTable)
<引数>	説明：テーブルデータ 取得方法：イベント内包 メソッド/値：イベント対象データ

取得した検索結果を参照用テーブルに設定する②

接続先コンポーネント②	■ テーブル (ID:20-11, KEY:登録用)
起動メソッド	テーブルデータを設定する (PFObjectTable)
<引数>	説明: テーブルデータ 取得方法: イベント内包 メソッド/値: イベント対象データ

確認 接続関係と配置は以下のようになります。

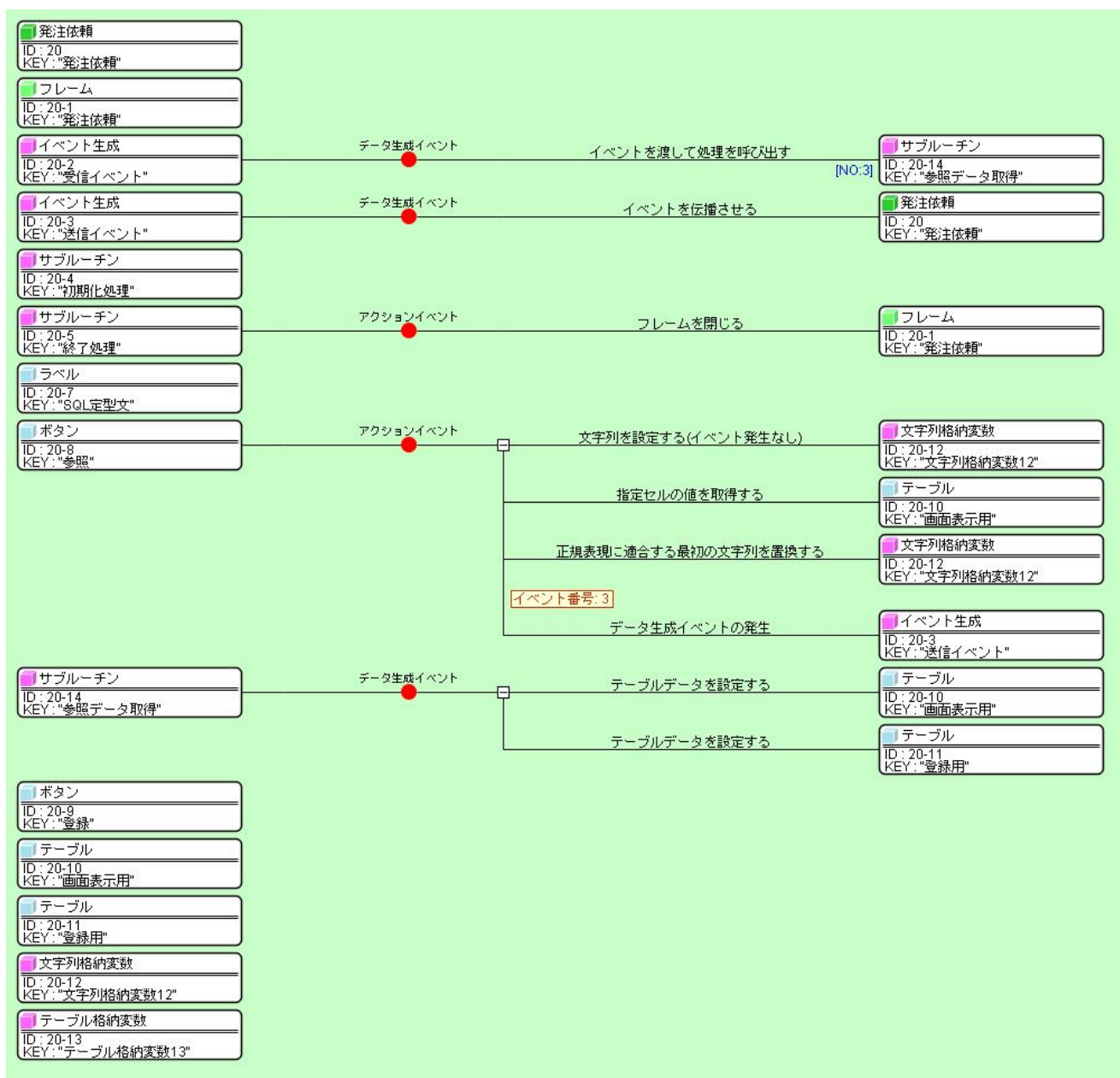


図 59 参照機能作成と登録機能作成準備(コンポーネント接続図)

操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 配置方法を「手動配置」に切り替えます。
- ③ 画面編集画面で「ボタン」コンポーネントおよび「テーブル」コンポーネントをフレームに追加します。

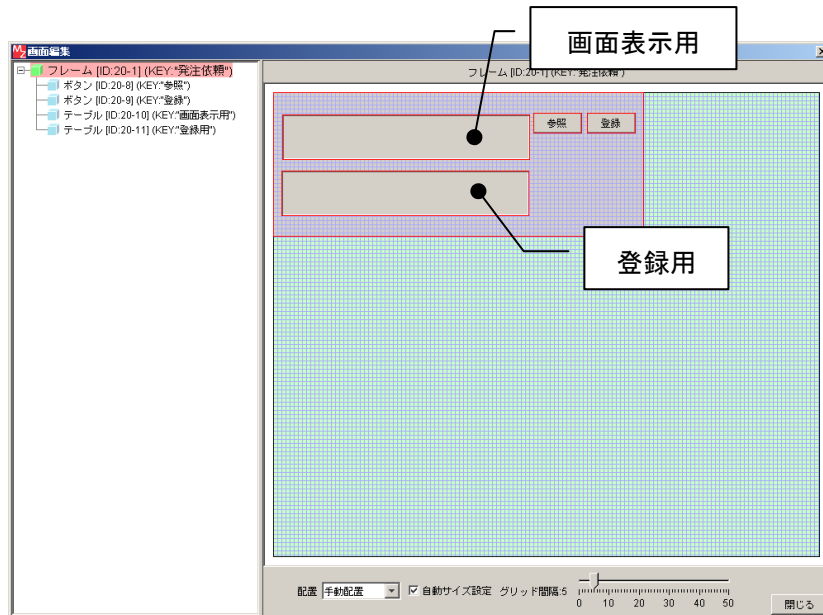


図 60 参照機能作成と登録機能作成準備(画面配置)

- ④ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

操 作

テーブルに検索結果を設定しましょう。

- ① [実行(設定可)]ボタンをクリックし、アプリケーションを起動します。
- ② [接続]ボタンをクリックし、「kobai」データベースに接続します。
- ③ メニューから、[購買管理] - [発注依頼...] と選択し、[発注依頼]フレームを表示します。
- ④ 画面表示用テーブル上で右クリックし、[テーブル] - [テーブル作成...] と選択し、1行1列のテーブルを作成します。列型は `java.lang.String` のままとします。

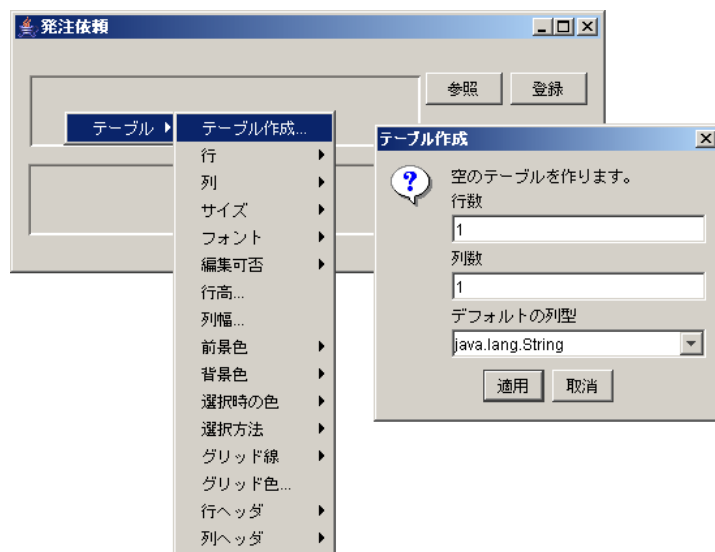


図 61 画面表示用テーブルの作成

- ⑤ [参照]ボタンをクリックします。検索結果が2つのテーブルに設定されます。

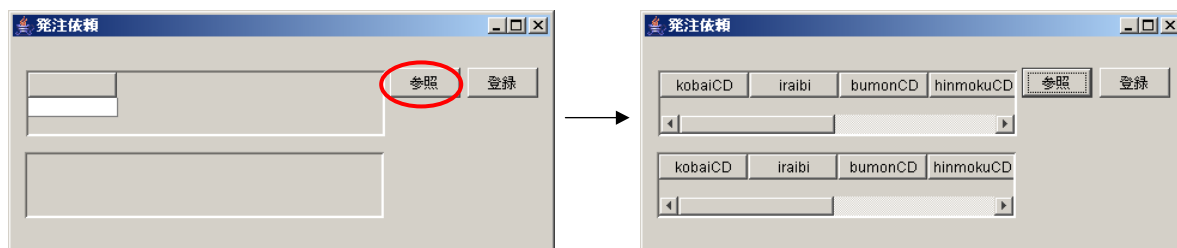


図 62 検索結果の設定

画面表示用のテーブルの列名を適切な日本語に設定し、列幅を調整します。また、iraibi(注文日)の表示形式を日付のみに設定します。さらに、画面表示用と登録用の2つのテーブルに1行を追加します。

操 作 画面表示用テーブルを編集しましょう。

- ① 画面表示用の列名を下の表に示すように日本語で設定します。

変更する列名上を右クリック－[列]－[列名...]をクリックします。

変更前	kobaiCD	iraibi	bumonCD	hinmokuCD	kiboSuryo	kiboTanka	kiboKingaku	shonin
変更後	注文番号	注文日	発注部門 コード	購買品 コード	発注数量	単価	金額	部門承認

- ② それぞれの列幅を調整します。

- ③ iraibi(注文日)の列型を日付に変更します。

列名上を右クリック－[列]－[列型]－[日付]をクリックします。

- ④ iraibi(注文日)の表示形式を日付のみに設定します。

列名上を右クリック－[列]－[表示形式(日付)]－[表示形式]－[日付のみ]をクリックします。



図 63 テーブルの列名および表示属性設定と行追加

- ⑤ 画面表示用と登録用のテーブルに1行追加します。

テーブル上を右クリック－[テーブル]－[行]－[追加]－[一行]とをクリックします。

- ⑥ 画面表示用と登録用のテーブルの shonin(部門承認)の列型を論理に変更します。

列名上を右クリック－[列]－[列型]－[論理]をクリックします。

画面編集画面でレイアウトを整えます。また、登録用テーブルは画面から削除しておきます。

操 作

画面表示用テーブルのレイアウトを整え、登録用テーブルを削除しましょう。

① アプリケーションを終了します。

② **画面編集** ボタンをクリックします。

③ レイアウトを整えます。

④ 登録用テーブルを削除します。

登録用テーブル上を右クリック → [コンポーネント削除] とクリックします。

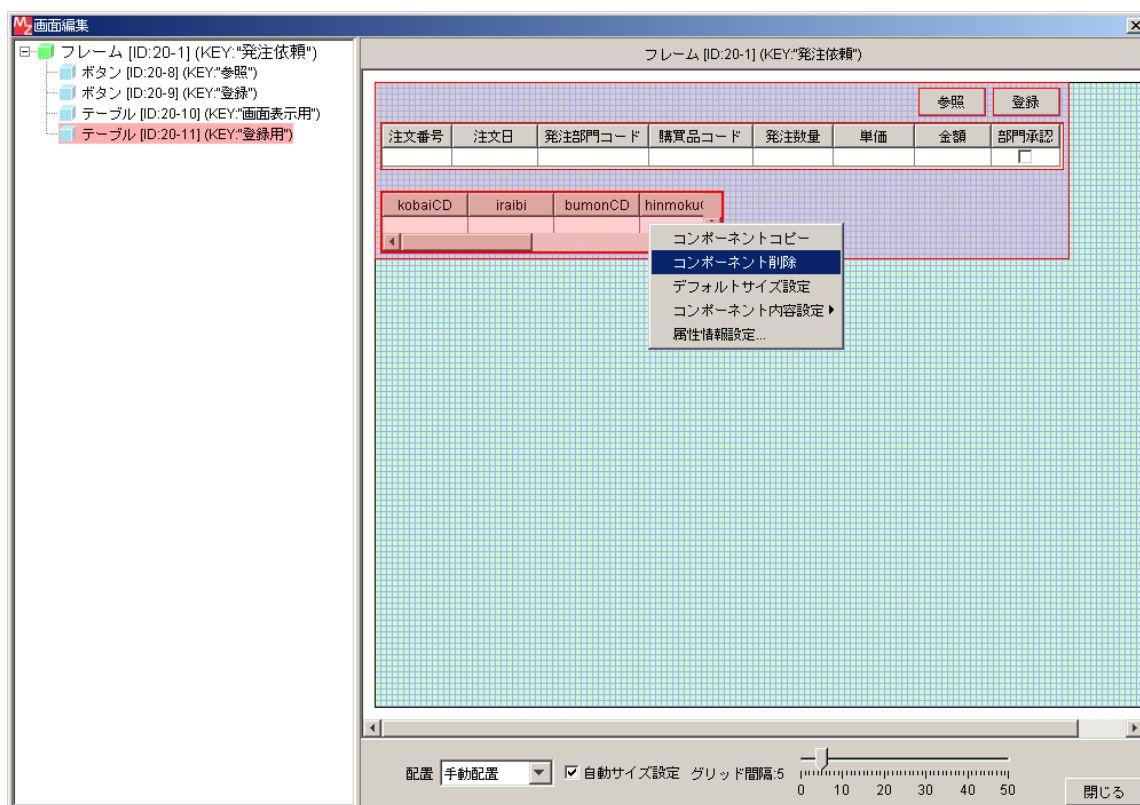


図 64 画面レイアウトの調整

⑤ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

参照データ取得後のサブルーチンの処理を修正します。¹²

追加するコンポーネント

- 比較演算(>)

修正する機能

1. 参照データ取得サブルーチンのうち、画面表示用と登録用の2つのテーブルに検索結果を設定する処理を削除する。
2. データ生成イベントの対象データ(検索結果)をテーブル格納変数に設定し、
3. その行数が0より大きければ、第0行のデータを画面表示用テーブルの第0行に設定する。

操 作

コンポーネントを追加し、接続しましょう。

- ① 「発注依頼」複合コンポーネントへ移動します。
- ② 「サブルーチン(参照データ取得)」コンポーネントから接続されているメソッドをいったん全て削除します。アクションイベント上で右クリック－[イベント処理削除]とクリックします。
- ③ 必要なコンポーネントを追加します。
作業領域で右クリック－[コンポーネント追加]－[処理部品]－[条件制御]－[比較演算(>)]とクリックします。

¹² 購買管理 12.mzax

④ 下の表に示すようにコンポーネントを接続します。

データ生成イベントのイベント対象データ(検索結果)をテーブル格納変数に設定する①

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ サブルーチン (ID:20-14, KEY:参照データ取得)
発生イベント	データ生成イベント
接続先コンポーネント①	■ テーブル格納変数 (ID:20-13)
起動メソッド	テーブルを設定する (PFObjectTable)
<引数>	説明: 設定するテーブル 取得方法: イベント内包 メソッド/値: イベント対象データ

テーブル格納変数に格納されているテーブルの行数が0より大きいかを演算する②

接続先コンポーネント②	■ 比較演算(>) (ID:20-15)
起動メソッド	数値に変換して左右オペランドに設定した後で 演算を行う (String, String)
<引数 0>	説明: 左オペランド 取得方法: メソッド戻り値 コンポーネント: テーブル格納変数 (ID:20-13) メソッド/値: 行数を取得する
<引数 1>	説明: 右オペランド 取得方法: 固定値 メソッド/値: 0

行数が 0 より大きければ(演算結果が true ならば)第 0 行のデータを取得する ❶

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ 比較演算(>) (ID:20-15)
発生イベント	処理完了イベント
接続先コンポーネント ❶	■ テーブル格納変数 (ID:20-13)
起動メソッド	行データリストを位置指定で取得する(int)
<引数>	説明：指定位置 取得方法：固定値 メソッド／値：0
イベント番号	1

テーブル格納変数に格納されている第 0 行のデータを画面表示用テーブルの第 0 行に設定する ❷

接続先コンポーネント ❷	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	指定行に行データをリスト形式で設定する (PFObjectList, int)
<引数 0>	取得方法：メソッド処理結果 メソッド／値：行データリストを位置指定で取得する
<引数 1>	取得方法：固定値 メソッド／値：0
イベント番号	1

確認

接続関係と配置は以下のようになります。

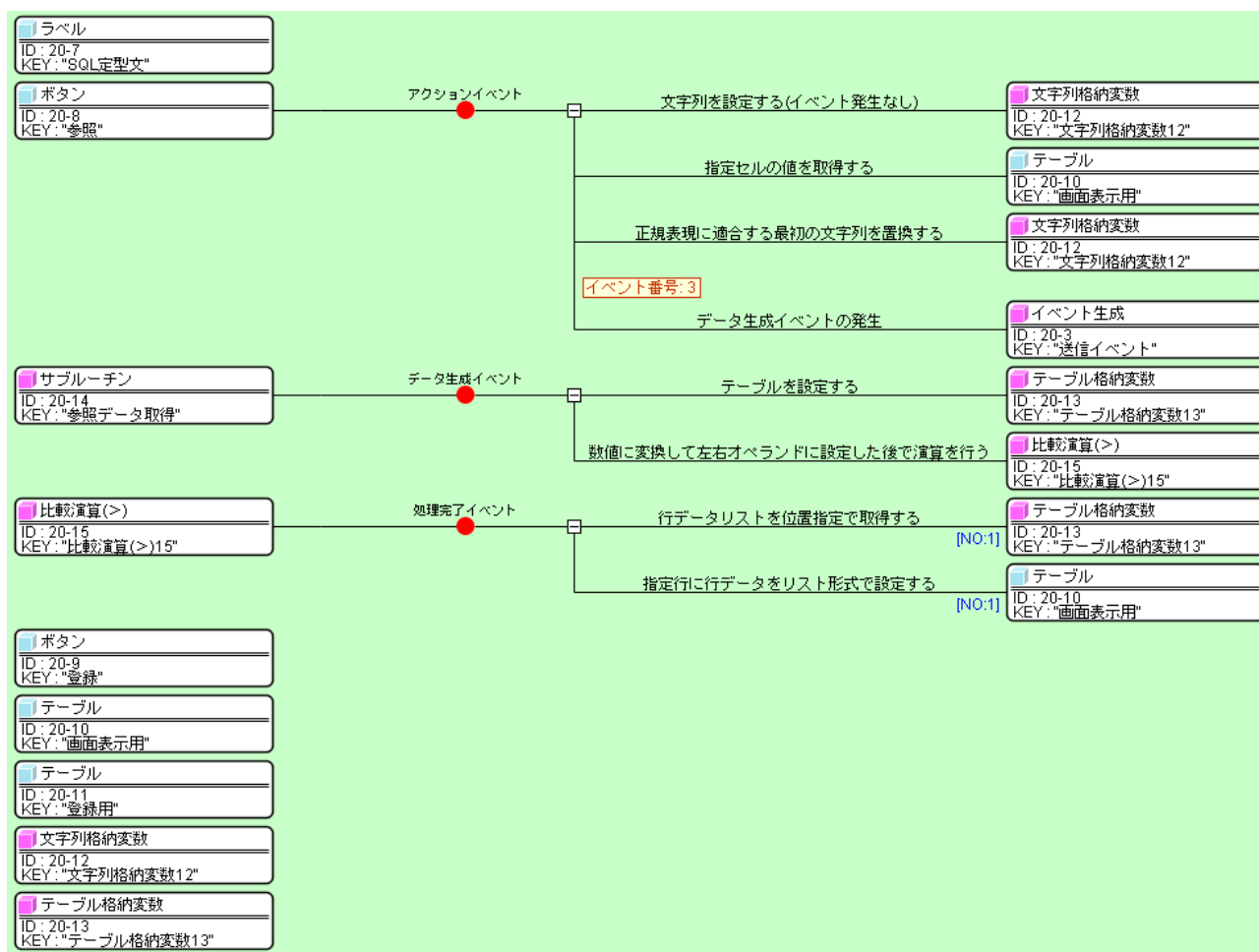


図 65 参照データ取得後の処理の修正

3.8. 登録機能の作成

発注依頼の登録には、INSERT文(SQL文)を利用しても構いませんが、ここでは「ローカルMySQL管理」複合コンポーネントが持つ「指定したテーブルに複数行を一括追加する(String, PFObjectTable)」メソッドを使うことにします。¹³

追加する機能

1. [登録]ボタンをクリックすると、
2. 画面表示用テーブルの行データを登録用テーブルの第0行に設定し、
3. データ更新イベントを使ってテーブル名(irai)とテーブルデータを送信する。
4. トップ階層で、「発注依頼」複合コンポーネントから発生するデータ更新イベントを「ローカル MySQL 管理」複合コンポーネントに接続し、「指定したテーブルに複数行を一括追加する(String, PFObjectTable)」を起動する。

¹³ 購買管理 13.mzax

操 作

コンポーネントを接続しましょう。

- ① 下の表に示すようにコンポーネントを接続します。

登録ボタンをクリックすると画面表示用テーブルの行のデータを取得する①

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ボタン (ID:20-9, KEY:登録)
発生イベント	アクションイベント
接続先コンポーネント①	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	指定行の行データをリスト形式で取得する(int)
<引数>	取得方法: 固定値 メソッド/値: 0

取得した画面表示用テーブルの行のデータを登録用テーブルの第0行に設定する②

接続先コンポーネント②	■ テーブル (ID:20-11, KEY:登録用)
起動メソッド	指定行に行データをリスト形式で設定する (PFObjectList, int)
<引数0>	取得方法: メソッド処理結果 メソッド/値: 指定行の行データをリスト形式で取得する
<引数1>	取得方法: 固定値 メソッド/値: 0

データ更新イベントでテーブル名(irai)とテーブルデータを送信する③

接続先コンポーネント③	■ イベント生成 (ID:20-3, KEY:送信イベント)
起動メソッド	データ更新イベントの発生(Object, Object, int)
<引数0>	説明: 対象データ 取得方法: 固定値 メソッド/値: irai
<引数1>	説明: 更新データ 取得方法: メソッド戻り値 コンポーネント: テーブル(ID:20-11, KEY:登録用) メソッド/値: テーブルデータを取得する
<引数2>	説明: イベント番号 取得方法: 固定値 メソッド/値: 0

データ生成コンポーネントのデータ更新イベントを上位階層へ伝播する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ イベント生成 (ID:20-3, KEY:送信イベント)
発生イベント	データ更新イベント
接続先コンポーネント	■ 発注依頼 (ID:20, KEY:発注依頼)
起動メソッド	イベントを伝播させる (PFEvent)
<引数>	取得方法 : イベント

確認

接続関係と配置は以下のようになります。

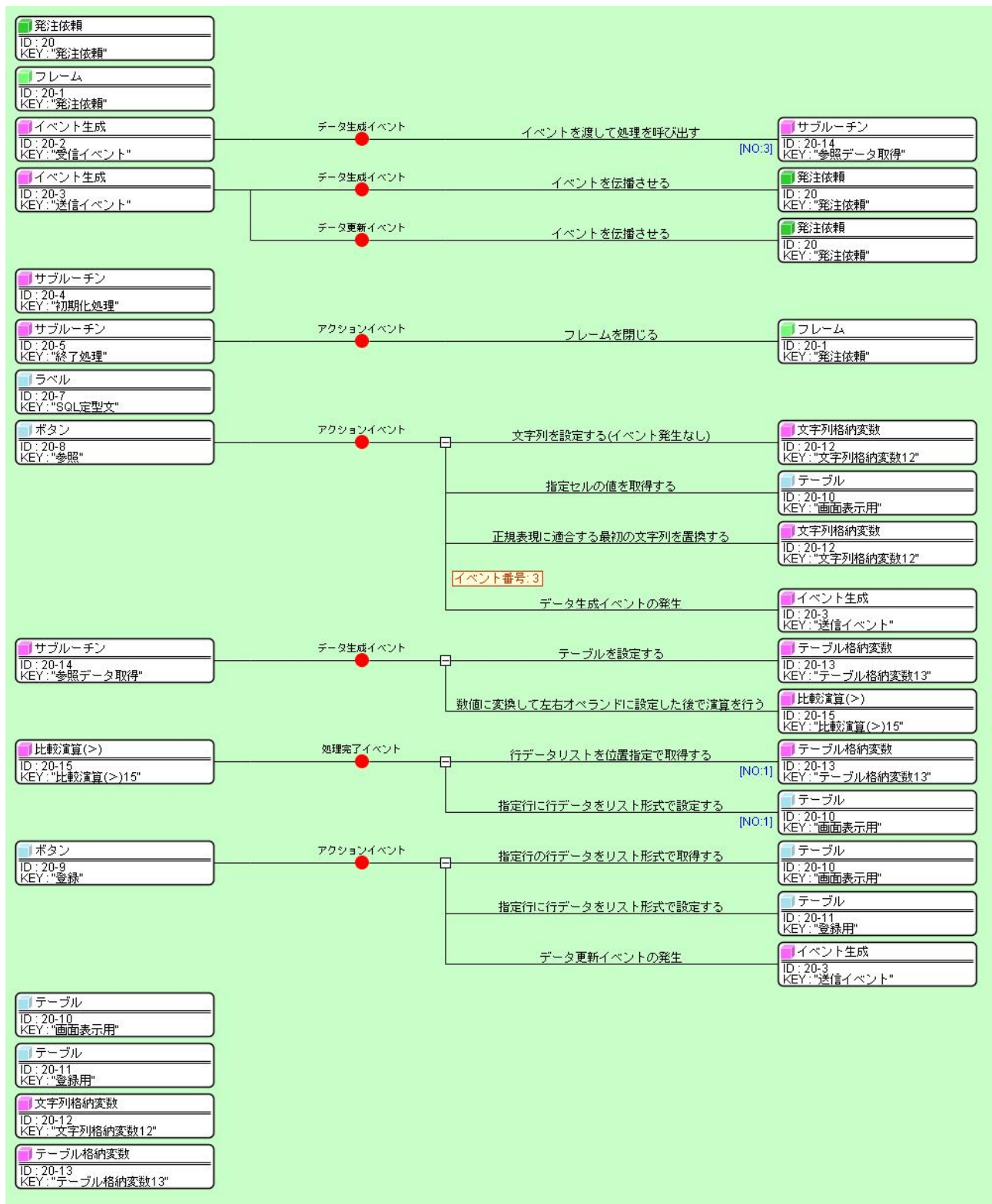


図 66 登録機能の作成(「発注依頼」複合コンポーネント内)

② アプリケーションビルダーのトップ画面に移動します。

③ 下の表に示すようにコンポーネントを接続します。

発注依頼複合コンポーネントのデータ更新イベントをローカル MySQL 管理複合コンポーネントに接続し
指定したテーブル(irai)に複数行を一括追加する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ 発注依頼 (ID:20, KEY:発注依頼)
発生イベント	データ更新イベント
接続先コンポーネント	■ ローカル MySQL 管理 (ID:2)
起動メソッド	指定したテーブルに複数行を一括追加する (String, PFObjectTable)
<引数 0>	取得方法：イベント内包 メソッド／値：イベント対象データ
<引数 1>	取得方法：イベント内包 メソッド／値：更新データ

確 認 接続関係と配置は以下のようになります。

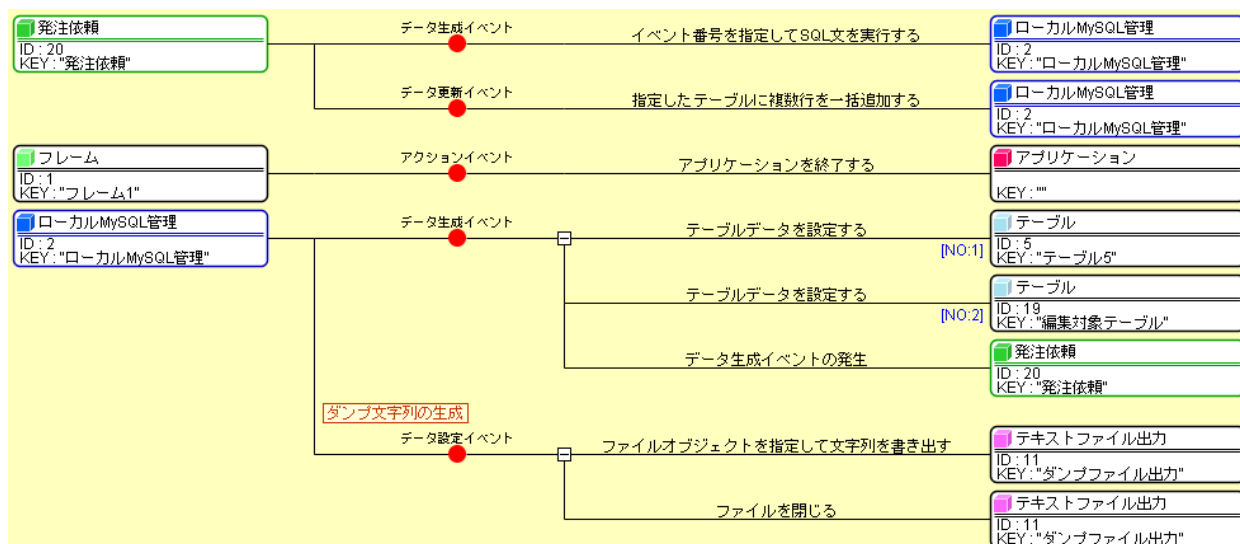


図 67 登録機能の作成(トップ階層)

操 作 アプリケーションの動作を確認しましょう。

- ① **実行**もしくは**実行(設定可)**ボタンをクリックし、アプリケーションを起動します。
- ② **[接続]**ボタンをクリックし、「kobai」データベースに接続します。
- ③ メニューから、**[購買管理]** - **[発注依頼...]** とクリックします。
- ④ 適当なデータを入力して**[登録]**ボタンをクリックします。

注文番号	注文日	発注部門コード	購買品コード	発注数量	単価	金額	部門承認
A1	2005/12/20	S1	P1	10	10,000	100,000	☐

図 68 登録機能の動作確認 1

- ⑤ メインウィンドウで、「select * from irai」という SQL 文を実行し、登録がされていることを確認します。

- ⑥ 確認後、アプリケーションを終了します。

3.9. 削除機能の作成

レコードを削除するときは、以下のDELETE定型文を用意し、「KOBACD」を削除対象の注文番号で置き換えます。¹⁴

```
delete from irai where kobaiCD='KOBACD'
```

追加するコンポーネント

- ラベル
- ボタン(「ボタン(参照)」コンポーネントをコピーして使用)

追加する機能

1. [削除]ボタンをクリックすると、
2. 文字列格納変数コンポーネントに DELETE 定型文を設定し、
3. 「KOBACD」を画面表示用テーブルの 0 行 0 列のセル値で置き換え、
4. その SQL 文をイベント番号 0 で実行する。

操 作	コンポーネントを追加し、接続しましょう。
------------	----------------------

- ① 「発注依頼」複合コンポーネントへ移動します。
- ② 必要なコンポーネントを追加します。
作業領域で右クリック- [コンポーネント追加] - [画面構成部品] - [テキスト] - [ラベル]
とクリックします。
- ③ [ラベル] コンポーネントに属性情報を設定します。属性情報画面を表示し、
[text]を「delete from irai where kobaiCD='KOBACD'」、[ComponentKey]を「DELETE 定型文」
と設定します。
- ④ 「ボタン(参照)」コンポーネントをコピー&ペーストします。
ボタンを右クリック- [コンポーネントコピー] し、作業領域で右クリック- [コンポーネント
ペースト] をクリックします。
- ⑤ ボタン名を変更します。属性情報設定画面を表示し、[text]を「削除」と設定します。

¹⁴ 購買管理 14.mzax

⑥ 下の表に示すようにコンポーネントを接続します。

削除ボタンをクリックすると文字列格納変数に DELETE SQL 定型文を設定する ❶

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ボタン (ID:20-17, KEY:削除)
発生イベント	アクションイベント
接続先コンポーネント ❶	■ 文字列格納変数 (ID:20-12)
起動メソッド	文字列を設定する(イベント発生なし) (String)
<引数>	取得方法: メソッド戻り値 コンポーネント: ラベル (ID:20-16, KEY:DELETE 定型文) メソッド/値: ラベルのテキスト文字列を取得する

画面表示用テーブルの 0 行 0 列の値を返す ❷

接続先コンポーネント ❷	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	指定セルの値を取得する (int, int)
<引数 0>	説明: 指定行 取得方法: 固定値 メソッド/値: 0
<引数 1>	説明: 指定列 取得方法: 固定値 メソッド/値: 0

KOBAICD を画面表示用テーブルの 0 行 0 列の値に置き換える ③

接続先コンポーネント ③	■ 文字列格納変数 (ID:20-12)
起動メソッド	正規表現に適合する最初の文字列を置換する (String, String)
<引数 0>	取得方法：固定値 メソッド／値：KOBAICD
<引数 1>	取得方法：メソッド処理結果 メソッド／値：指定セルの値を取得する

SQL 文をイベント番号 0 で実行する ④

接続先コンポーネント ④	■ イベント生成 (ID:20-3, KEY:送信イベント)
起動メソッド	データ生成イベントの発生 (Object, int)
<引数 0>	説明：生成データ 取得方法：メソッド戻り値 コンポーネント：文字列格納変数 (ID:20-12) メソッド／値：文字列を取得する
<引数 1>	説明：イベント番号 取得方法：固定値 メソッド／値：0

確 認 接続関係と配置は以下のようになります。

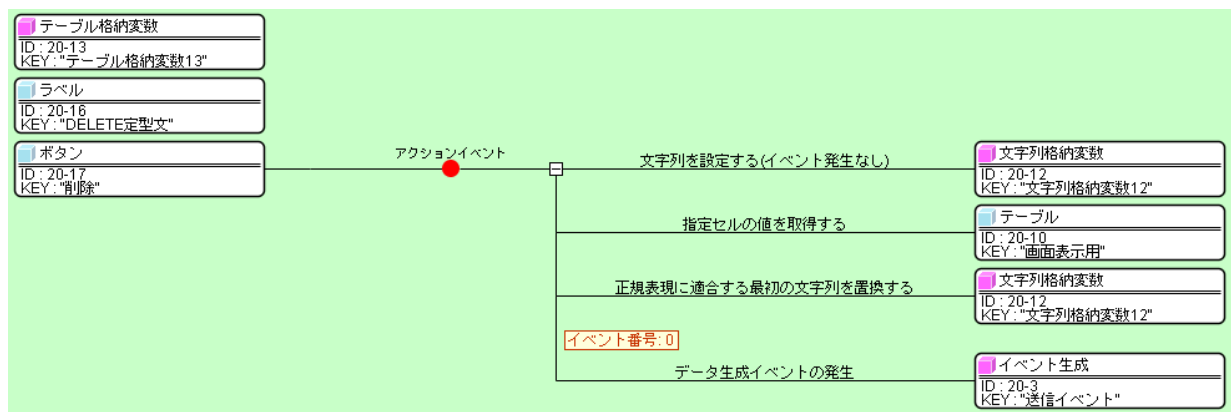


図 69 削除機能の作成(コンポーネント接続関係)

操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 画面編集画面で「ボタン」コンポーネントをフレームに追加します。

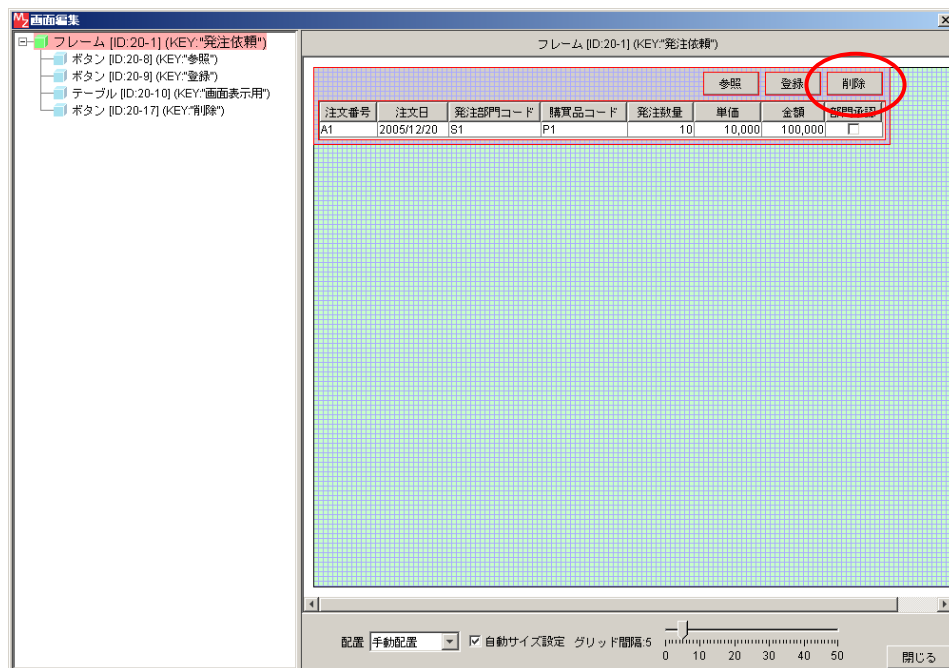


図 70 削除機能の作成(画面配置)

- ③ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

操 作 アプリケーションの動作を確認しましょう。

- ① アプリケーションビルダーのトップ画面に移動します。
- ② **実行**もしくは**実行(設定可)**ボタンをクリックし、アプリケーションを起動します。
- ③ **[接続]**ボタンをクリックし、「kobai」データベースに接続します。
- ④ メニューから、**[購買管理]** - **[発注依頼...]** とクリックします。
- ⑤ **[削除]**ボタンをクリックします。



図 71 削除機能の動作確認 1

- ⑥ メインウィンドウで、「select * from irai」という SQL 文を実行し、レコードが削除されていることを確認します。

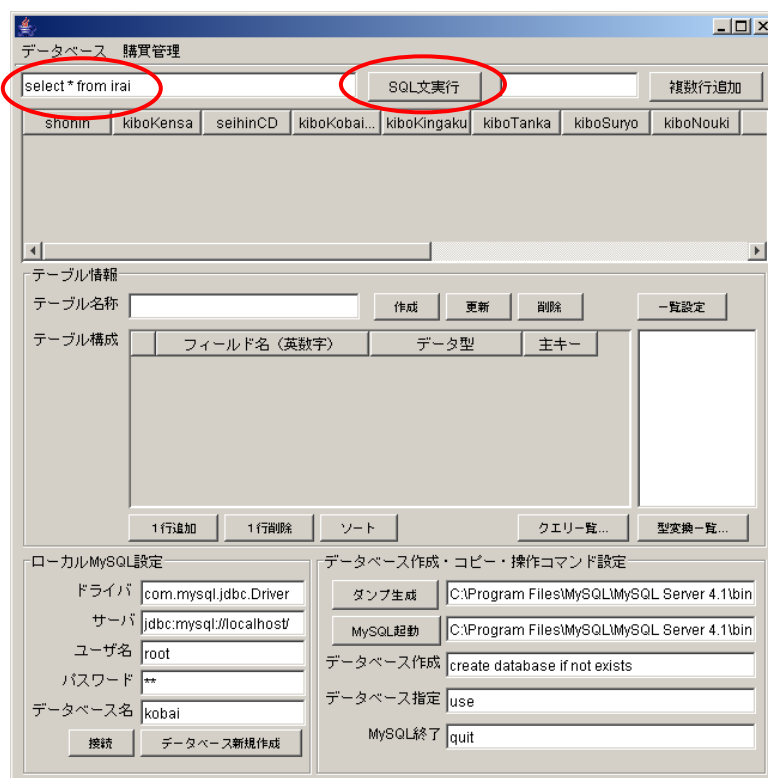


図 72 削除機能の動作確認 2

- ⑦ 確認後、アプリケーションを終了します。

3. 10. 更新機能の作成(登録機能の修正)

レコードの更新には、UPDATE文を利用しても構いませんが、ここでは該当するレコードを一度削除し、あらためて更新後のデータでレコードを登録するようにします。¹⁵

追加するコンポーネント

- サブルーチン

追加する機能

1. [登録]ボタンをクリックすると、
2. 画面表示用テーブルの0行0列の kobaiCD を持つレコードを irai テーブルから削除し、
3. 画面表示用テーブルのデータを irai テーブルへ登録する。

すなわち、すでに作成した登録機能の最初の部分に削除機能を追加することになります。そのために、[削除]ボタンに接続している処理をサブルーチンとしてまとめ、そのサブルーチンを[登録]ボタンから起動するようにします。

操 作	コンポーネントを追加し、接続しましょう。
------------	----------------------

- ① 「発注依頼」複合コンポーネントへ移動します。
- ② 必要なコンポーネントを追加します。
作業領域で右クリック - [コンポーネント追加] - [処理部品] - [サブルーチン] - [サブルーチン] とクリックします。
- ③ コンポーネントキーを設定します。属性情報設定画面を表示し、[ComponentKey]を「削除処理」と設定します。
- ④ 「サブルーチン(削除処理)」コンポーネントにアクションイベントを追加します。
- ⑤ 「ボタン(削除)」コンポーネントから接続されているメソッドを全てコピーします。
アクションイベント上で右クリック - [起動メソッドコピー] とクリックします。
- ⑥ 「サブルーチン(削除処理)」コンポーネントにメソッドを貼り付けます。
アクションイベント上で右クリック - [起動メソッド貼り付け] とクリックします。
- ⑦ 「ボタン(削除)」コンポーネントから接続されているメソッドを全て削除します。
アクションイベント上で右クリック - [イベント処理削除] とクリックします。

¹⁵ 購買管理 15.mzax

- ⑧ 下の表に示すようにコンポーネントを接続します。

削除ボタンをクリックすると削除処理サブルーチンを呼び出す

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)s	■ ボタン (ID:20-17, KEY:削除)
発生イベント	アクションイベント
接続先コンポーネント	■ サブルーチン (ID:20-18, KEY:削除処理)
起動メソッド	処理を呼び出す()

登録ボタンをクリックすると削除処理サブルーチンを呼び出す

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ボタン (ID:20-9, KEY:登録)
発生イベント	アクションイベント
接続先コンポーネント	■ サブルーチン (ID:20-18, KEY:削除処理)
起動メソッド	処理を呼び出す()

- ⑨ 「ボタン(登録)」コンポーネントのアクションイベントから接続されている起動メソッドの順番を入れ替えます。「サブルーチン(削除処理)」メソッドを1番目にします。

確認

接続関係と配置は以下のようになります。

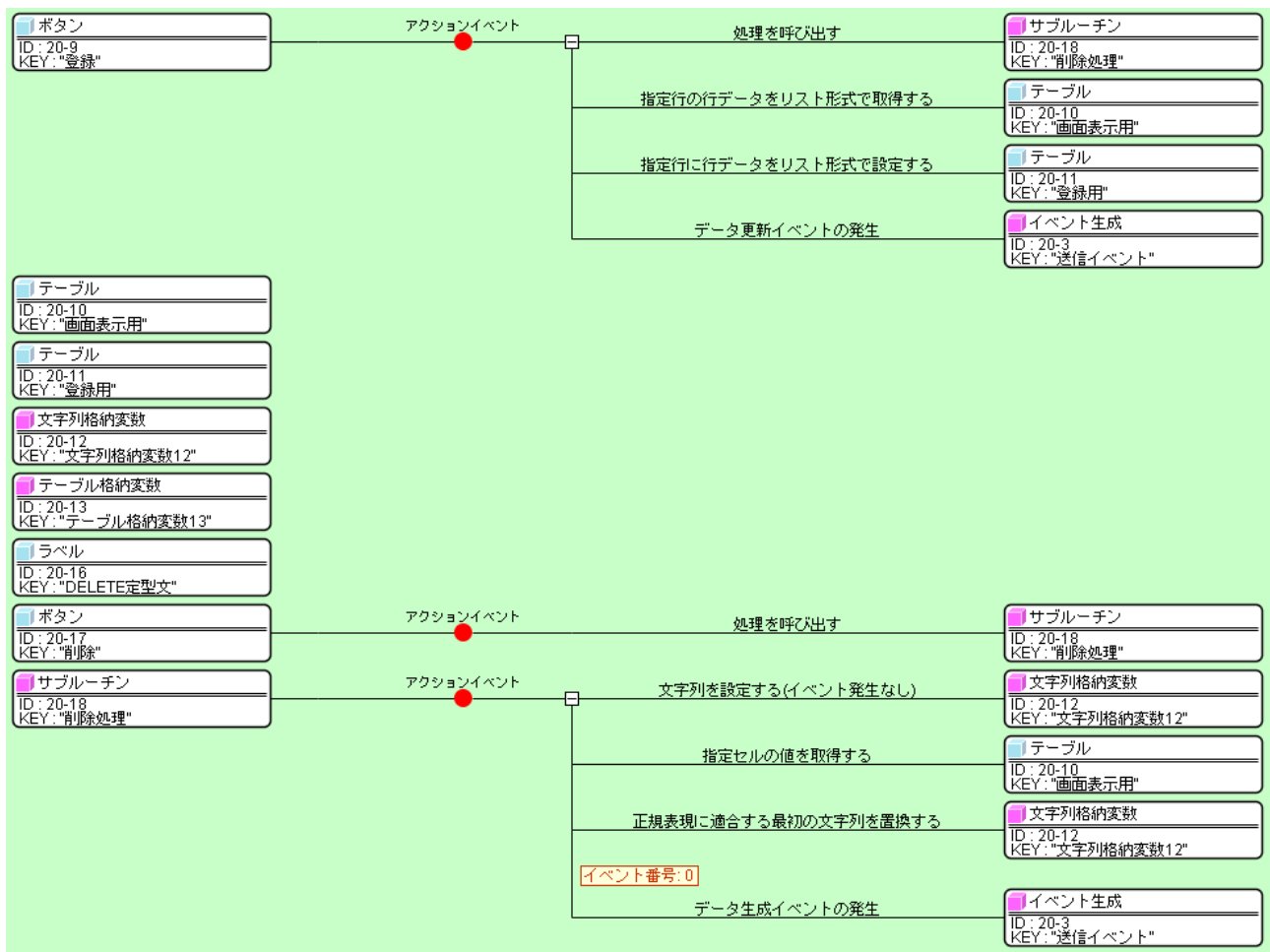


図 73 更新機能の作成(登録機能の修正)

操作

アプリケーションの動作を確認しましょう。

- ① アプリケーションビルダーのトップ画面に移動します。
- ② **実行**もしくは**実行(設定可)**ボタンをクリックし、アプリケーションを起動します。
- ③ [接続]ボタンをクリックし、「kobai」データベースに接続します。
- ④ メニューから、[購買管理] - [発注依頼...] とクリックします。
- ⑤ アプリケーションを実行し、更新機能の動作確認を行います。
- ⑥ 確認後、アプリケーションを終了します。

3.11. 注文番号一覧設定機能の作成

登録済みの発注依頼を参照するとき、注文番号の一覧が設定されていると便利です。そこで、画面表示用テーブルの第0列に、登録済み注文番号のリストを設定できるようにします。¹⁶

追加するコンポーネント

- ラベル
- ボタン
- サブルーチン

追加する機能

1. [注文番号一覧設定] ボタンをクリックすると、
2. 注文番号一覧を取得するための SQL 文をイベント番号 4 で実行し、
3. 取得した結果を画面表示用テーブルの第0列の選択リストに設定する。
4. 終了時に、設定した選択リストを削除する。

操 作	コンポーネントを追加し、接続しましょう。
------------	----------------------

- ① 「発注依頼」複合コンポーネントへ移動します。
- ② 必要なコンポーネントを追加します。
作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [テキスト] - [ラベル]
とクリックします。
作業領域で右クリック - [コンポーネント追加] - [画面構成部品] - [ボタン] - [ボタン]
とクリックします。
作業領域で右クリック - [コンポーネント追加] - [処理部品] - [サブルーチン] -
[サブルーチン] とクリックします。
- ③ 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。
[ラベル] コンポーネントの[text]を「select kobaiCD from irai order by kobaiCD」、
[ComponentKey]を「注文番号一覧取得」、
[ボタン] コンポーネントの[text]を「注文番号一覧設定」、
[サブルーチン]コンポーネントの[ComponentKey]を「注文番号一覧取得」
と設定します。

¹⁶ 購買管理 16.mzax

- ④ 下の表に示すようにコンポーネントを接続します。

注文番号一覧設定ボタンをクリックすると注文番号一覧を取得する SQL 文をイベント番号 4 で実行する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ボタン (ID:20-20, KEY:注文番号一覧設定)
発生イベント	アクションイベント
接続先コンポーネント	■ イベント生成 (ID:20-3, KEY:送信イベント)
起動メソッド	データ生成イベントの発生 (Object, int)
<引数 0>	説明: 生成データ 取得方法: メソッド戻り値 コンポーネント: ラベル (ID:20-19, KEY:注文番号一覧取得) メソッド/値: ラベルのテキスト文字列を取得する
<引数 1>	説明: イベント番号 取得方法: 固定値 メソッド/値: 4

イベントを注文番号一覧取得サブルーチンに渡して処理を呼び出す

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ イベント生成 (ID:20-2, KEY:受信イベント)
発生イベント	データ生成イベント
接続先コンポーネント	■ サブルーチン (ID:20-21, KEY:注文番号一覧取得)
起動メソッド	イベントを渡して処理を呼び出す (PFEvent)
<引数>	説明: イベント 取得方法: イベント
イベント番号	4

データ生成イベントのイベント対象データ(検索結果)をテーブル格納変数に設定する❶

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ サブルーチン (ID:20-21, KEY:注文番号一覧取得)
発生イベント	データ生成イベント
接続先コンポーネント❶	■ テーブル格納変数 (ID:20-13)
起動メソッド	テーブルを設定する (PFObjectTable)
<引数>	説明: 設定するテーブル 取得方法: イベント内包 メソッド/値: イベント対象データ

テーブル格納変数に格納されているデータの第0列をリスト形式で取得する❷

接続先コンポーネント❷	■ テーブル格納変数 (ID:20-13)
起動メソッド	列データリストを位置指定で取得する (int)
<引数>	取得方法: 固定値 メソッド/値: 0

取得した結果を画面表示用テーブルの第0列の選択リストに設定する❸

接続先コンポーネント❸	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	指定列に選択項目リストを設定する (PFObjectList, int)
<引数 0>	説明: 選択項目リスト 取得方法: メソッド処理結果 メソッド/値: 列データリストを位置指定で取得する
<引数 1>	説明: 指定列 取得方法: 固定値 メソッド/値: 0

終了時に設定した選択リストを削除する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ サブルーチン (ID:20-5, KEY:終了処理)
発生イベント	アクションイベント
接続先コンポーネント	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	指定列に選択項目リストを設定する (PFObjectList, int)
<引数 0>	説明: 選択項目リスト 取得方法: 固定値 メソッド/値: なし
<引数 1>	説明: 指定列 取得方法: 固定値 メソッド/値: 0

- ⑦ 「サブルーチン(終了処理)」コンポーネントのアクションイベントから接続されている起動メソッドの順番を入れ替えます。「テーブル(画面表示用)」のメソッドを上に配置します。

確認 接続関係と配置は以下のようになります。

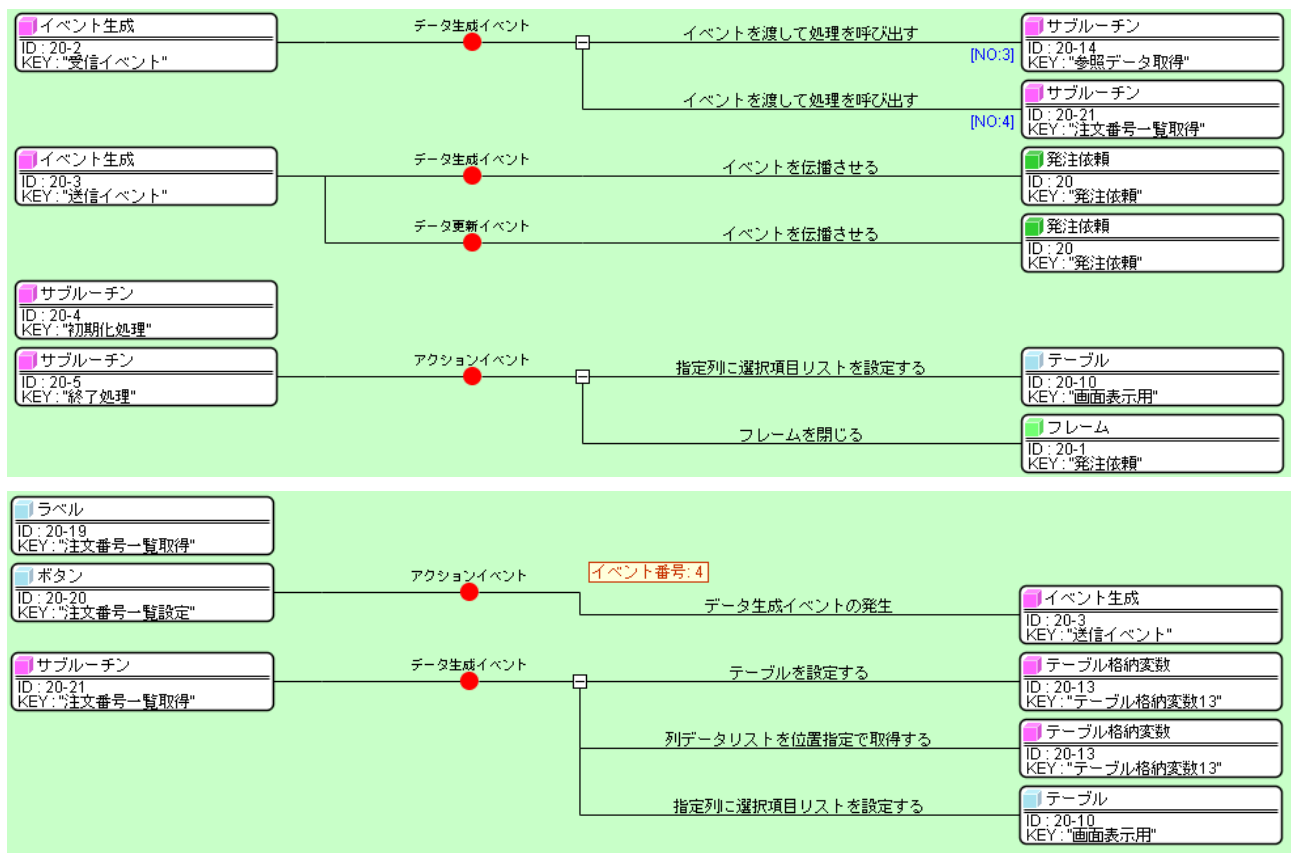


図 74 注文番号一覧設定機能の作成(コンポーネント接続関係)

操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 画面編集画面で「ボタン」コンポーネントをフレームに追加します。

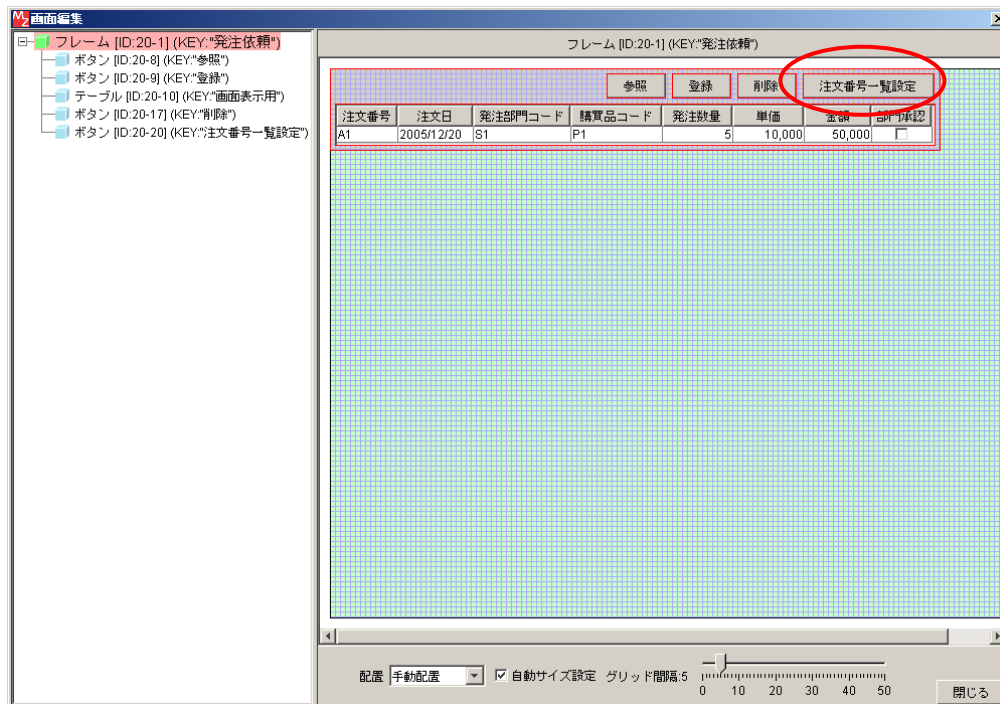


図 75 注文番号一覧設定機能の作成(画面配置)

- ③ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

操 作

アプリケーションの動作を確認しましょう。

- ① アプリケーションビルダーのトップ画面に移動します。
- ② **実行**もしくは**実行(設定可)**ボタンをクリックし、アプリケーションを起動します。
- ③ **[接続]**ボタンをクリックし、「kobai」データベースに接続します。
- ④ メニューから、**[購買管理]** - **[発注依頼...]** とクリックします。
- ⑤ **[注文番号一覧設定]**ボタンをクリックします。
- ⑥ **[注文番号]**のセルをクリックし、選択リストが設定されることを確認します。



図 76 注文番号選択リストの設定

- ⑦ 確認後、アプリケーションを終了します。

3.12. 金額計算機能の作成

発注数量と単価を入力したとき、自動的に金額を計算する機能を作成します。¹⁷

追加するコンポーネント

- イベント生成
- サブルーチン
- 乗算

追加する機能

1. 画面表示用テーブルのセルが更新されたとき、
2. 画面表示用テーブルの第4列あるいは第5列が選択されていれば、
3. 画面表示用テーブルの選択状態を解除し、
4. 第0行4列のセル値と第0行5列のセル値の積を第0行6列のセルに設定する。

!!! 注意事項 !!!

乗算実行結果を第0行6列のセルに設定する前に、画面表示用テーブルの選択状態を解除する設定を忘れると無限ループに陥ります。

操 作	コンポーネントを追加し、接続しましょう。
------------	----------------------

- ① 「発注依頼」複合コンポーネントへ移動します。
- ② 必要なコンポーネントを追加します。
作業領域で右クリック - [コンポーネント追加] - [処理部品] - [イベント] - [イベント生成] とクリックします。
作業領域で右クリック - [コンポーネント追加] - [処理部品] - [サブルーチン] - [サブルーチン] とクリックします。
作業領域で右クリック - [コンポーネント追加] - [処理部品] - [演算制御] - [乗算(×)] とクリックします。
- ③ 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。
[イベント生成] コンポーネントの[ComponentKey]を「セル値更新」、
[サブルーチン]コンポーネントの[ComponentKey]を「発注数量×単価の計算」と設定します。

¹⁷ 購買管理 17.mzax

- ④ 下の表に示すようにコンポーネントを接続します。

画面表示用テーブルのセルが更新されたとき更新されたセルの列インデックスを伝達する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ テーブル (ID:20-10, KEY:画面表示用)
発生イベント	データ更新イベント
接続先コンポーネント	■ イベント生成 (ID:20-23, KEY:セル値更新)
起動メソッド	アクションイベントの発生(int)
<引数>	説明: イベント番号 取得方法: メソッド戻り値 コンポーネント: テーブル (ID:20-10, KEY:画面表示用) メソッド/値: 選択列の位置を取得する
イベント番号	0

発注数量×単価の計算サブルーチンを呼び出す

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ イベント生成 (ID:20-23, KEY:セル値更新)
発生イベント	アクションイベント
接続先コンポーネント	■ サブルーチン (ID:20-24, KEY:発注数量×単価の計算)
起動メソッド	処理を呼び出す()
イベント番号	4、5

画面表示用テーブルの選択状態を解除する①

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ サブルーチン (ID:20-24, KEY:発注数量×単価の計算)
発生イベント	アクションイベント
接続先コンポーネント①	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	clearSelection()

第0行4列のセル値を取得する②

接続先コンポーネント②	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	指定セルの値を取得する(int, int)
<引数0>	説明: 指定行 取得方法: 固定値 メソッド/値: 0
<引数1>	説明: 指定列 取得方法: 固定値 メソッド/値: 4

第0行5列のセル値を取得する③

接続先コンポーネント③	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	指定セルの値を取得する(int, int)
<引数0>	説明: 指定行 取得方法: 固定値 メソッド/値: 0
<引数1>	説明: 指定列 取得方法: 固定値 メソッド/値: 5

発注数量(第 0 行 4 列) × 単価(第 0 行 5 列)を計算する ④

接続先コンポーネント ④	■ 乗算(×) (ID:20-22)
起動メソッド	数値変換/左右オペランド設定後、演算を行う (String, String)
<引数 0>	説明：左オペランド 取得方法：メソッド処理結果 メソッド/値：指定セルの値を取得する(第 0 行 4 列)
<引数 1>	説明：右オペランド 取得方法：メソッド処理結果 メソッド/値：指定セルの値を取得する(第 0 行 5 列)

乗算実行結果を第 0 行 6 列のセルに設定する

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ 乗算(×) (ID:20-22)
発生イベント	処理完了イベント
接続先コンポーネント	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	セルの値を設定する(Object, int, int)
<引数 0>	説明：セルの値 取得方法：イベント内包 メソッド/値：処理結果データ
<引数 1>	説明：行の位置 取得方法：固定値 メソッド/値：0
<引数 2>	説明：列の位置 取得方法：固定値 メソッド/値：6
イベント番号	0

確認 接続関係と配置は以下のようになります。

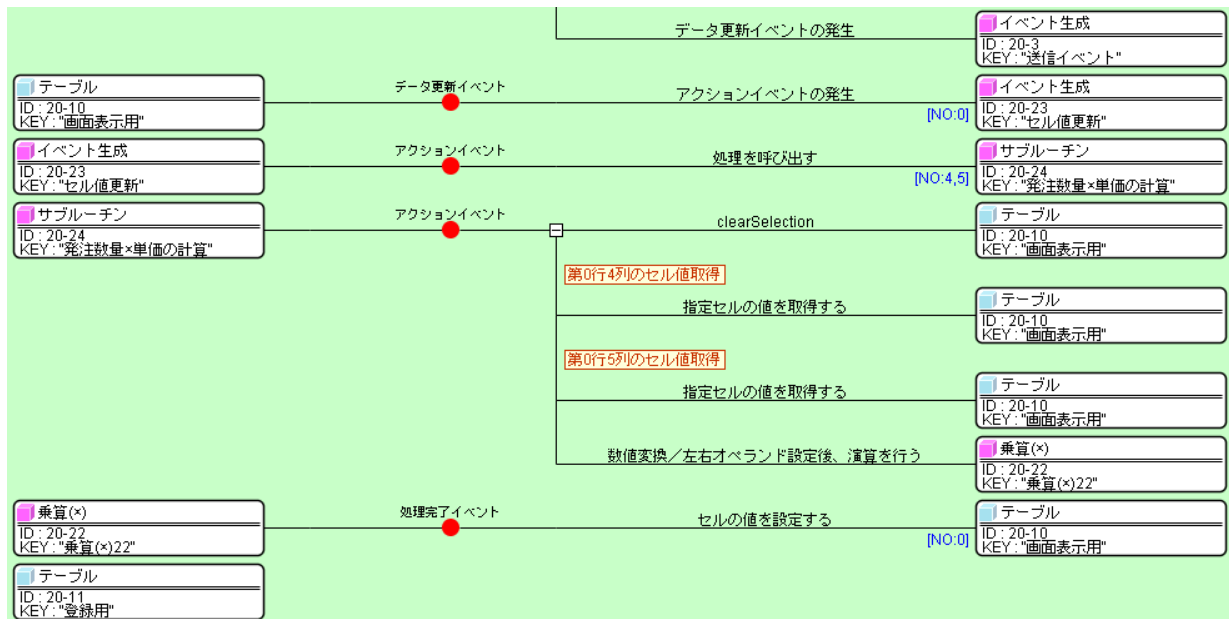


図 77 金額計算機能の作成

操作 アプリケーションの動作を確認しましょう。

- ① アプリケーションビルダーのトップ画面に移動します。
- ② **実行**もしくは**実行(設定可)**ボタンをクリックし、アプリケーションを起動します。
- ③ **[接続]**ボタンをクリックし、「kobai」データベースに接続します。
- ④ メニューから、**[購買管理]**－**[発注依頼...]** とクリックします。
- ⑤ 金額計算機能の動作確認を行います。
- ⑥ 確認後、アプリケーションを終了します。

3.13. 伝票発行機能の作成

下図のような伝票を発行する機能を作成します。¹⁸

発注依頼書			
注文番号	A1	単価	10000.0
発注部門コード	S1	発注数量	5
購買品コード	P1	金額	50000

注文日	部門承認
2005/12/20 00:00:00	FALSE

図 78 伝票発行

追加するコンポーネント

- ラベル(注文番号用)
- ラベル(発注部門コード用)
- ラベル(購買品コード用)
- ラベル(単価用)
- ラベル(発注数量用)
- 実数(Double)格納変数(金額用)
- ボタン
- サブルーチン(帳票用ラベル設定用)
- サブルーチン(帳票用テーブル作成用)
- オブジェクトキュー
- 帳票

追加する機能

1. [伝票発行...]ボタンをクリックすると、
2. 画面表示用テーブルのセルの値をラベルおよび実数(Double)格納変数コンポーネントに設定するとともに、
3. 印刷用のテーブル(注文日と部門承認欄用)を作成し、
4. それらのデータに基づく発注依頼書の印刷プレビューを表示する。

¹⁸ 購買管理 18.mzax

操 作

コンポーネントを追加し、接続しましょう。

① 「発注依頼」複合コンポーネントへ移動します。

② 必要なコンポーネントを一括追加します。

作業領域で右クリック- [コンポーネント一括追加...] とクリックします。

[画面構成部品] パネルが表示されていることを確認します。

右側の領域から [テキスト] をクリックし、[ラベル] の追加数に「5」と入力します。

右側の領域から [ボタン] をクリックし、[ボタン] の追加数に「1」と入力します。

[処理部品] パネルに切り替えます。

右側の領域から [変数] をクリックし、[浮動小数点数(Double)格納変数] の追加数に「1」と入力します。

右側の領域から [サブルーチン] をクリックし、[サブルーチン] の追加数に「2」と入力します。

右側の領域から [配列] をクリックし、[オブジェクトキュー] の追加数に「1」と入力します。

[入出力] パネルに切り替えます。

右側の領域から [帳票] をクリックし、[帳票] の追加数に「1」と入力します。

最後に[追加]ボタンをクリックします。

③ 属性情報を設定します。属性情報画面を表示し、それぞれ次のように設定します。

注文番号用として追加した [ラベル] コンポーネントの[ComponentKey]を「注文番号」、

発注部門コード用として追加した [ラベル] コンポーネントの[ComponentKey]を「発注部門コード」、

購買品コード用として追加した [ラベル] コンポーネントの[ComponentKey]を「購買品コード」、

単価用として追加した [ラベル] コンポーネントの[ComponentKey]を「単価」、

発注数量用として追加した [ラベル] コンポーネントの[ComponentKey]を「発注数量」、

[ボタン] コンポーネントの[text]を「伝票発行...」、

帳票用ラベル設定用として追加した [サブルーチン] コンポーネントの[ComponentKey]を

「帳票用ラベル設定」、

帳票用テーブル作成用として追加した [サブルーチン] コンポーネントの[ComponentKey]を

「帳票用テーブル作成」

と設定します。

④ 下の表に示すようにコンポーネントを接続します。

伝票発行ボタンをクリックすると帳票用ラベル設定サブルーチンを呼び出す①

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ ボタン (ID:20-31, KEY:伝票発行...)
発生イベント	アクションイベント
接続先コンポーネント①	■ サブルーチン (ID:20-33, KEY:帳票用ラベル設定)
起動メソッド	処理を呼び出す()

伝票発行ボタンをクリックすると帳票用テーブル作成サブルーチンを呼び出す②

接続先コンポーネント②	■ サブルーチン (ID:20-34, KEY:帳票用テーブル作成)
起動メソッド	処理を呼び出す()

伝票発行ボタンをクリックすると印刷プレビュー画面を表示する③

接続先コンポーネント③	■ 帳票 (ID:20-35)
起動メソッド	印刷プレビュー画面を表示する(Component)
<引数>	説明: 親コンポーネント 取得方法: 固定値 メソッド/値: なし

画面表示用テーブルの第 0 行のデータを取得する ❶

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■ サブルーチン (ID:20-33, KEY:帳票用ラベル設定)
発生イベント	アクションイベント
接続先コンポーネント ❶	■ テーブル (ID:20-10, KEY:画面表示用)
起動メソッド	指定行の行データをリスト形式で取得する(int)
<引数>	取得方法：固定値 メソッド／値：0

取得したデータをキューとして設定する ❷

接続先コンポーネント ❷	■ オブジェクトキュー (ID:20-32)
起動メソッド	リストによるキューの設定 (PFObjectList)
<引数>	取得方法：メソッド処理結果 メソッド／値：指定行の行データをリスト形式で取得する

ラベルに注文番号を設定する ❸

接続先コンポーネント ❸	■ ラベル (ID:20-25, KEY:注文番号)
起動メソッド	ラベルのテキスト文字列を設定する(String)
<引数>	説明：テキスト文字列 取得方法：メソッド戻り値 コンポーネント：オブジェクトキュー (ID:20-32) メソッド／値：オブジェクトの取得

オブジェクトを取得する④

接続先コンポーネント④	■オブジェクトキュー（ID:20-32）
起動メソッド	オブジェクトの取得()

ラベルに発注部門コードを設定する⑤

接続先コンポーネント⑤	■ラベル（ID:20-26, KEY:発注部門コード）
起動メソッド	ラベルのテキスト文字列を設定する(String)
<引数>	説明：テキスト文字列 取得方法：メソッド戻り値 コンポーネント：オブジェクトキュー（ID:20-32） メソッド/値：オブジェクトの取得

ラベルに購買品コードを設定する⑥

接続先コンポーネント⑥	■ラベル（ID:20-27, KEY:購買品コード）
起動メソッド	ラベルのテキスト文字列を設定する(String)
<引数>	説明：テキスト文字列 取得方法：メソッド戻り値 コンポーネント：オブジェクトキュー（ID:20-32） メソッド/値：オブジェクトの取得

ラベルに発注数量を設定する⑦

接続先コンポーネント⑦	■ラベル（ID:20-28, KEY:発注数量）
起動メソッド	ラベルのテキスト文字列を設定する(String)
<引数>	説明：テキスト文字列 取得方法：メソッド戻り値 コンポーネント：オブジェクトキュー（ID:20-32） メソッド/値：オブジェクトの取得

ラベルに単価を設定する⑧

接続先コンポーネント⑧	■ラベル (ID:20-29, KEY:単価)
起動メソッド	ラベルのテキスト文字列を設定する (String)
<引数>	説明: テキスト文字列 取得方法: メソッド戻り値 コンポーネント: オブジェクトキュー (ID:20-32) メソッド/値: オブジェクトの取得

ラベルに金額を設定する⑨

接続先コンポーネント⑨	■実数 (Double) 格納変数 (ID:20-36, KEY:金額)
起動メソッド	数値 (Double) を設定する (Double)
<引数>	説明: 設定する Double オブジェクト 取得方法: メソッド戻り値 コンポーネント: オブジェクトキュー (ID:20-32) メソッド/値: オブジェクトの取得

画面表示用テーブルデータをテーブル格納変数に設定する⑩

接続項目	接続関係
接続元コンポーネント (イベント発生コンポーネント)	■サブルーチン (ID:20-34, KEY:帳票用テーブル作成)
発生イベント	アクションイベント
接続先コンポーネント⑩	■テーブル格納変数 (ID:20-13)
起動メソッド	テーブルを設定する (PFObjectTable)
<引数>	説明: 設定するテーブル 取得方法: メソッド戻り値 コンポーネント: テーブル (ID:20-10, KEY:画面表示用) メソッド/値: テーブルデータを取得する

第 6 列を削除する ②

接続先コンポーネント ②	■ テーブル格納変数 (ID:20-13)
起動メソッド	列を位置指定で削除する (int)
< 引数 >	説明 : 指定位置 取得方法 : 固定値 メソッド / 値 : 6

第 5 列を削除する ③

接続先コンポーネント ③	■ テーブル格納変数 (ID:20-13)
起動メソッド	列を位置指定で削除する (int)
< 引数 >	説明 : 指定位置 取得方法 : 固定値 メソッド / 値 : 5

第 4 列を削除する ④

接続先コンポーネント ④	■ テーブル格納変数 (ID:20-13)
起動メソッド	列を位置指定で削除する (int)
< 引数 >	説明 : 指定位置 取得方法 : 固定値 メソッド / 値 : 4

第 3 列を削除する ⑤

接続先コンポーネント ⑤	■ テーブル格納変数 (ID:20-13)
起動メソッド	列を位置指定で削除する (int)
< 引数 >	説明 : 指定位置 取得方法 : 固定値 メソッド / 値 : 3

第 2 列を削除する ⑥

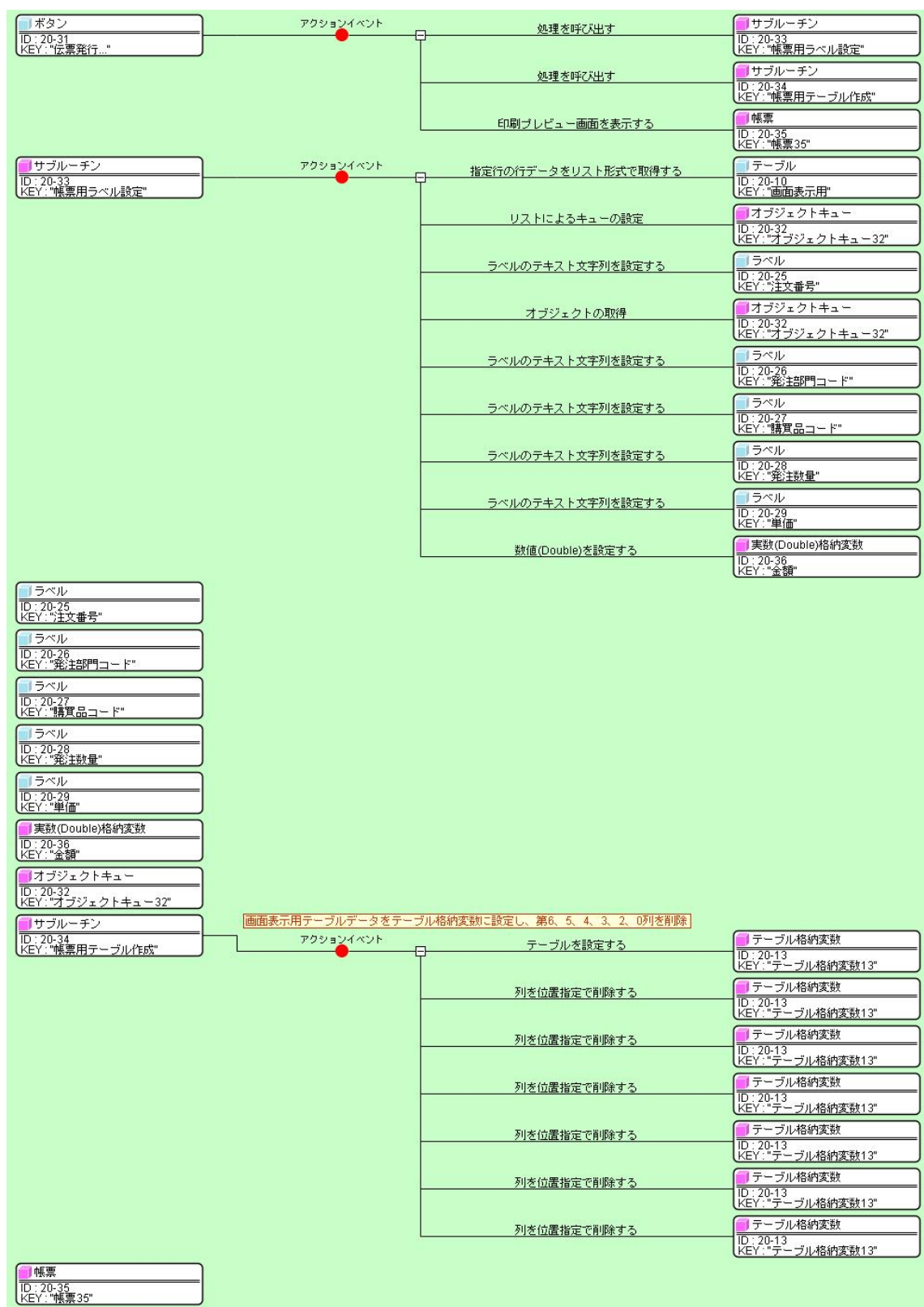
接続先コンポーネント ⑥	■ テーブル格納変数 (ID:20-13)
起動メソッド	列を位置指定で削除する (int)
< 引数 >	説明 : 指定位置 取得方法 : 固定値 メソッド / 値 : 2

第 0 列を削除する ⑦

接続先コンポーネント ⑦	■ テーブル格納変数 (ID:20-13)
起動メソッド	列を位置指定で削除する (int)
< 引数 >	説明 : 指定位置 取得方法 : 固定値 メソッド / 値 : 0

確認

接続関係と配置は以下のようになります。



操 作

画面を作成しましょう。

- ① **画面編集** ボタンをクリックします。
- ② 画面編集画面で「ボタン」コンポーネントをフレームに追加します。

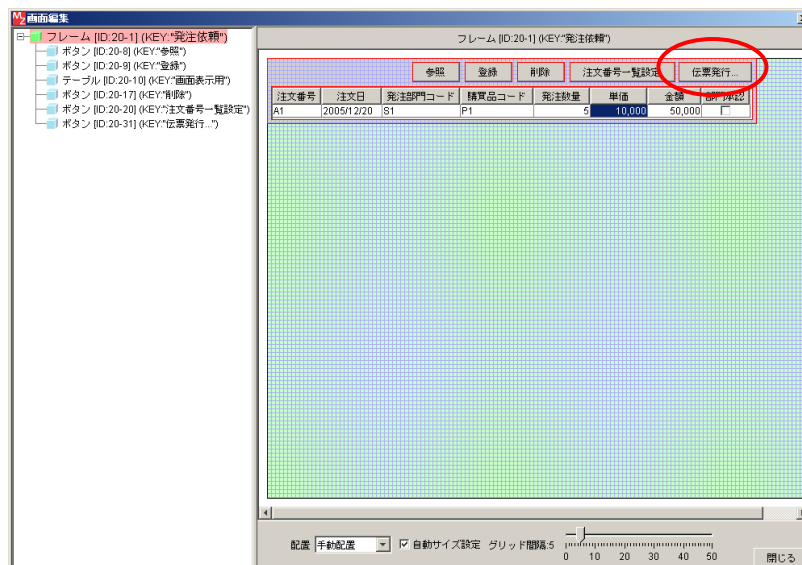


図 80 伝票発行機能の作成(画面配置)

- ③ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

操 作

伝票作成に使うデータを設定しておきましょう。

- ① **実行** もしくは **実行(設定可)** ボタンをクリックし、アプリケーションを起動します。
- ② メニューから、[購買管理] - [発注依頼...] とクリックします。
- ③ [伝票発行...] ボタンをクリックします。
空の [帳票プレビュー] ウィンドウが表示されます。
- ④ [帳票プレビュー] ウィンドウを閉じます。
- ⑤ アプリケーションを終了します。

操 作

印刷用の画面を作成しましょう。

- ① **帳票編集** ボタンをクリックします。
- ② 帳票サイズと帳票方向を設定します。
[用紙設定...] ボタンをクリックし、[A5 サイズ]、[横]に設定します。

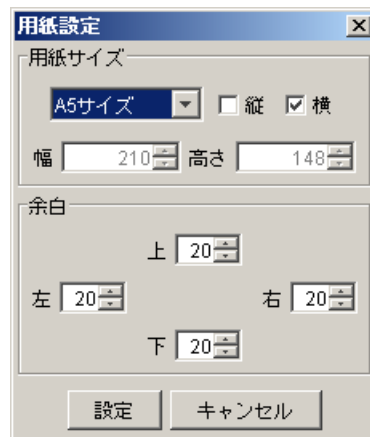


図 81 伝票発行機能の作成(用紙設定)

- ③ タイトル用のラベル要素をレイアウト画面に追加します。
レイアウト画面上で右クリック－[帳票要素追加]－[ラベル要素] とクリックします。
- ④ データを設定します。[ラベル要素] 上で右クリック－[データ設定]－[テキスト入力...] とクリックします。「発注依頼書」と入力します。
適当な場所へ移動し、大きさを調整します。
- ⑤ テーブル要素をレイアウト画面に追加します。
レイアウト画面上で右クリック－[帳票要素追加]－[テーブル要素] とクリックします。
- ⑥ データを設定します。[テーブル要素] 上で右クリック－[データ設定]－[データ取得設定...] とクリックします。[取得元コンポーネント] は [テーブル格納変数]、[データ取得メソッド] は [テーブルを取得する()] を選択します。
適当な場所へ移動し、大きさを調整します。

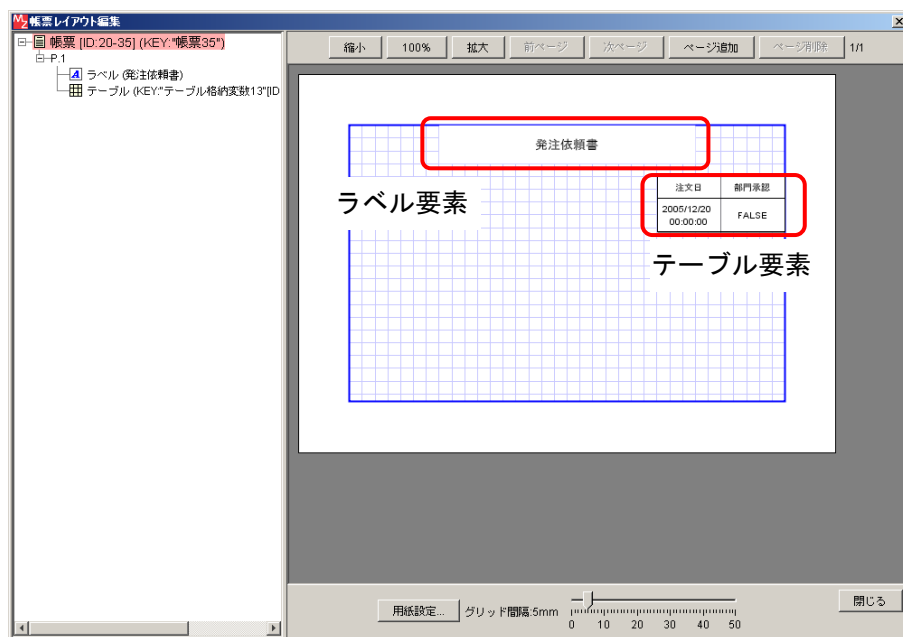
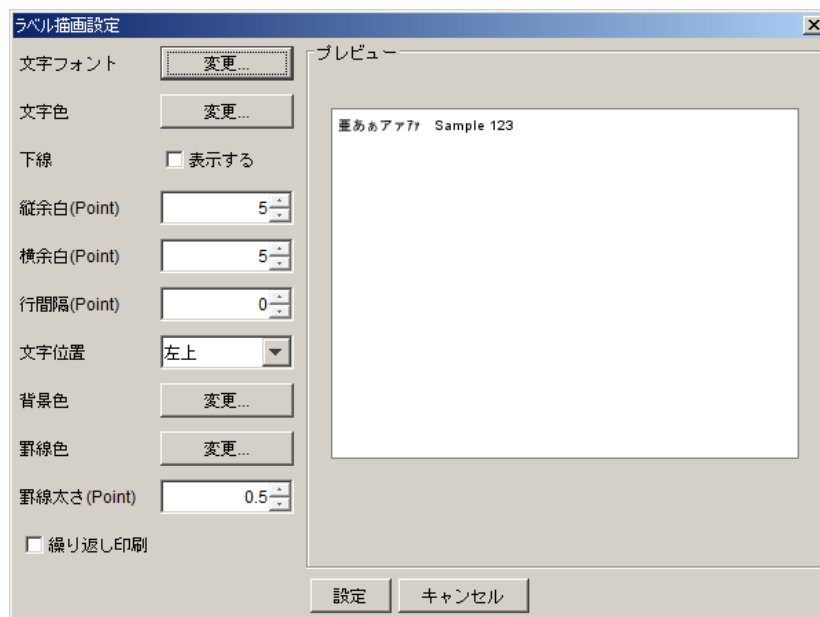


図 82 伝票発行機能の作成(画面配置 1)

ワンポイント – 描画属性設定 –

ラベル要素の文字フォントや文字色、文字表示位置、罫線色などの属性は、[ラベル描画設定] 画面で変更できます。

- ① レイアウト画面の [ラベル要素] 上で右クリック – [ラベル描画属性設定...] をクリックします。



- ② 任意の設定後 **設定** ボタンをクリックします。

⑦ ラベル要素をレイアウト画面に 12 個追加し、下の表に示すようにデータを設定します。

番号	データ設定	入力内容	取得元コンポーネント	データ取得メソッド
1	テキスト入力	注文番号		
2	テキスト入力	発注部門コード		
3	テキスト入力	購買品コード		
4	データ取得設定		ラベル(注文番号)	ラベルのテキスト文字列を取得する()
5	データ取得設定		ラベル(発注部門コード)	ラベルのテキスト文字列を取得する()
6	データ取得設定		ラベル(購買品コード)	ラベルのテキスト文字列を取得する()
7	テキスト入力	発注数量		
8	テキスト入力	単価		
9	テキスト入力	金額		
10	データ取得設定		ラベル(発注数量)	ラベルのテキスト文字列を取得する()
11	データ取得設定		ラベル(単価)	ラベルのテキスト文字列を取得する()
12	データ取得設定		実数(Double)格納変数 (金額)	long 値を取得する()

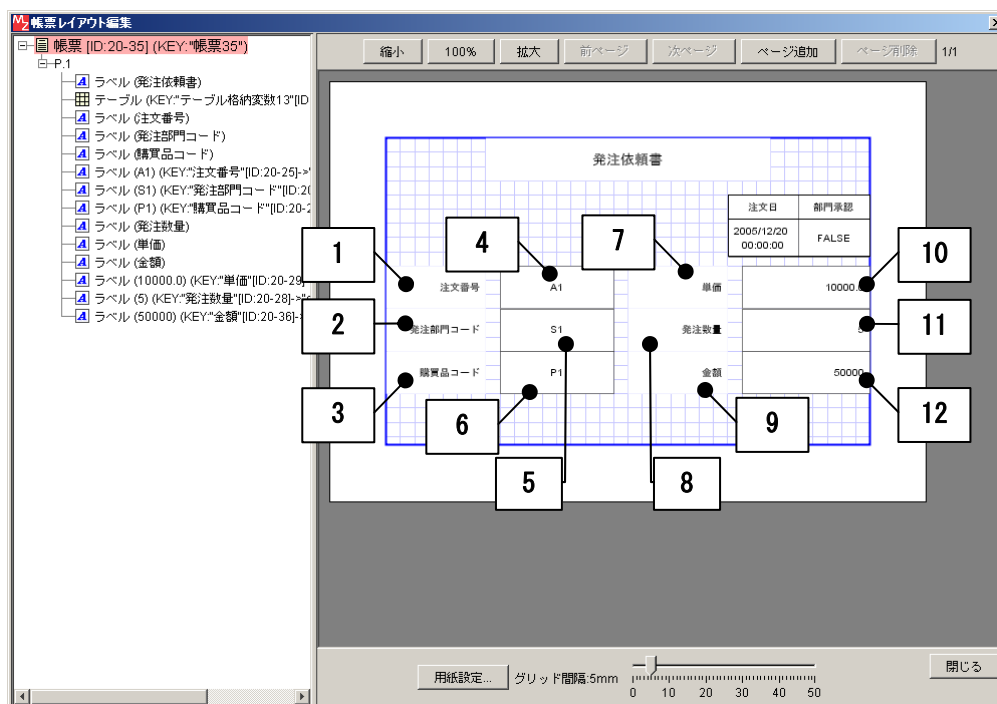


図 83 伝票発行機能の作成(画面配置 2)

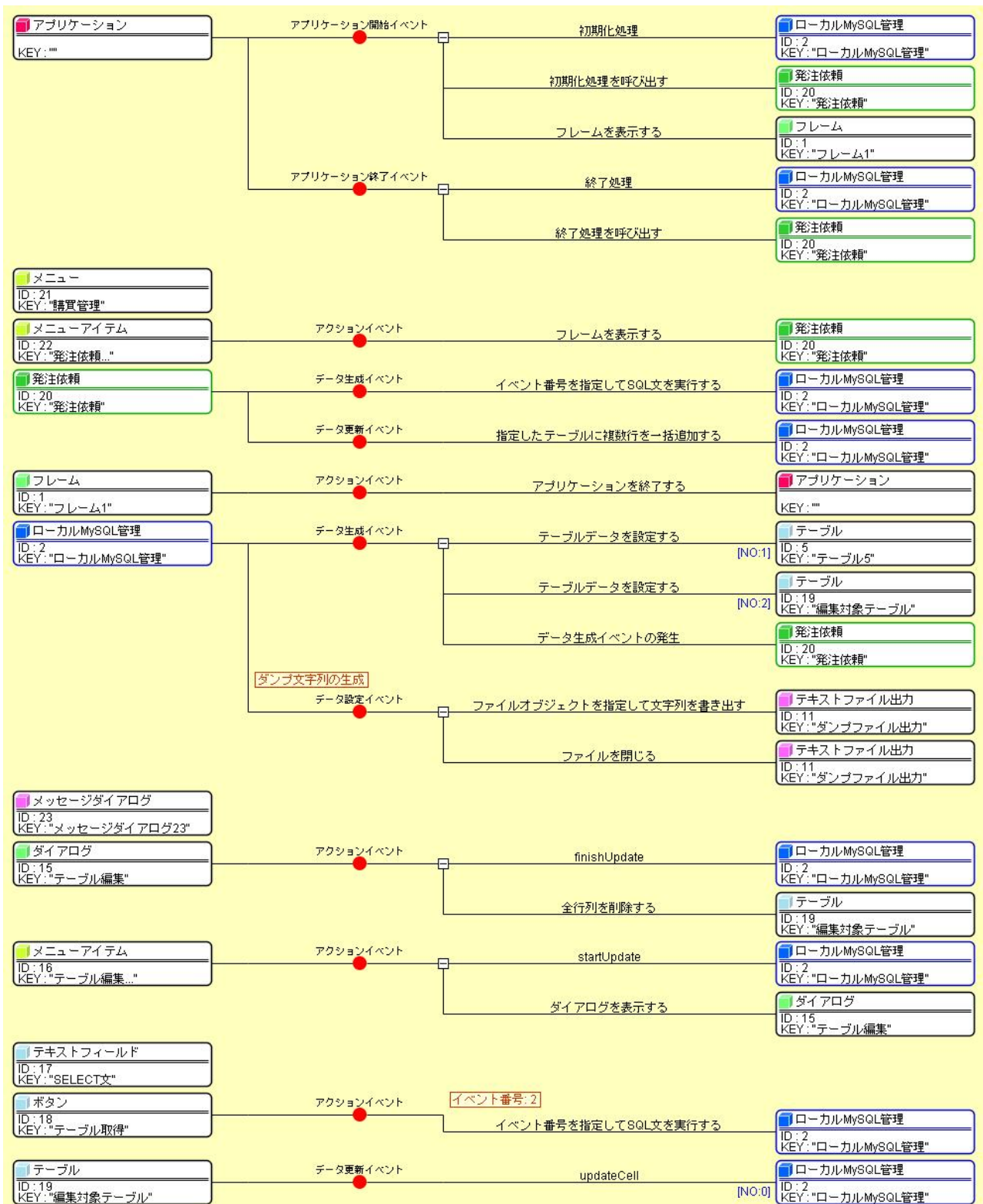
-
- ⑧ **閉じる** ボタンをクリックして、アプリケーションビルダー画面に戻ります。

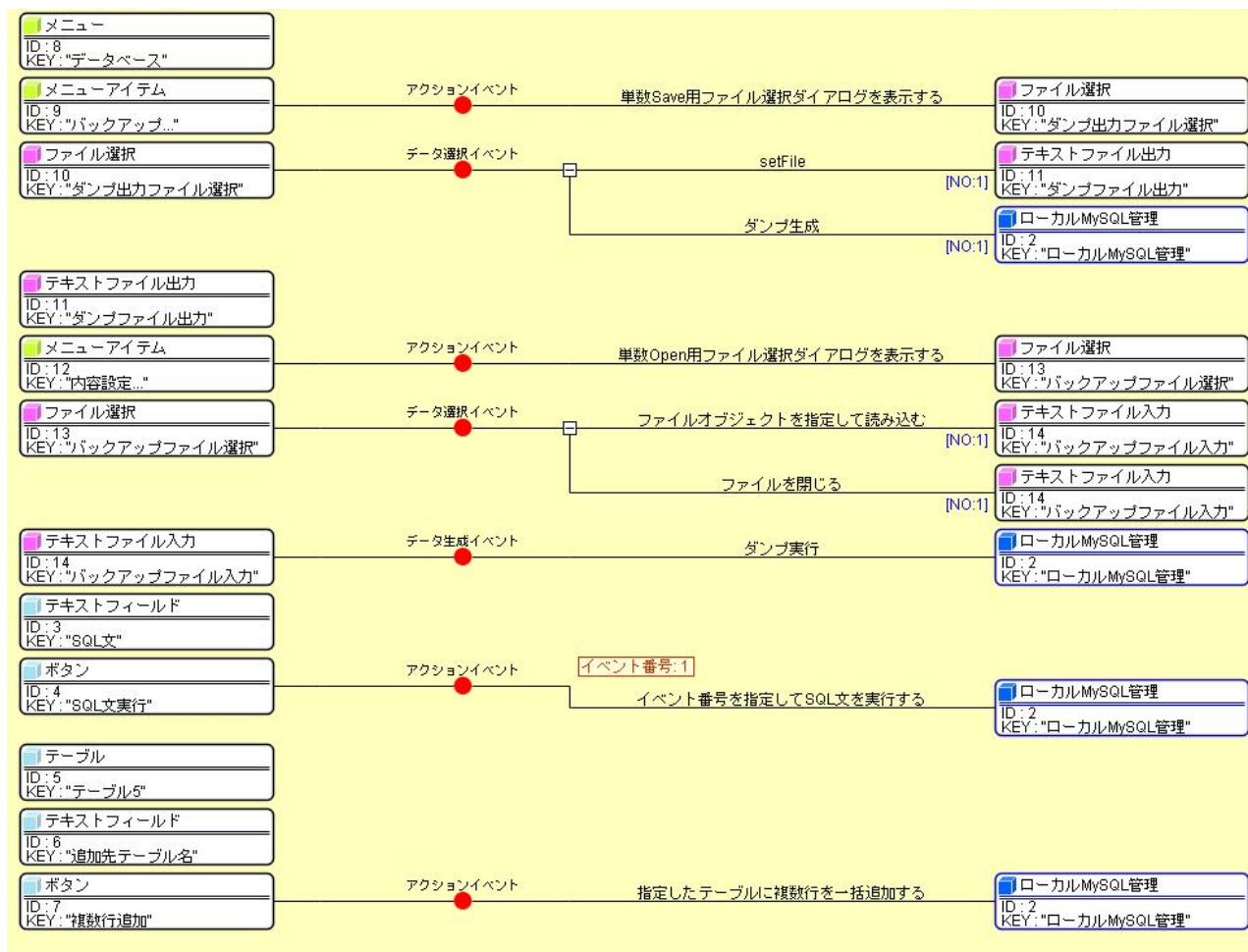
操 作 アプリケーションの動作を確認しましょう。

- ① アプリケーションビルダーのトップ画面に移動します。
- ② **実行** もしくは **実行(設定可)** ボタンをクリックし、アプリケーションを起動します。
- ③ **[接続]** ボタンをクリックし、「kobai」データベースに接続します。
- ④ メニューから、**[購買管理]** - **[発注依頼...]** とクリックします。
- ⑤ **[伝票発行...]** ボタンをクリックします。
- ⑥ 確認後、アプリケーションを終了します。

4. コンポーネント接続関係(まとめ)

トップ階層





「発注依頼」複合コンポーネント

