



新機能ガイド

Magic eDeveloper V9



Magic Software Japan K.K.

本マニュアルに記載の内容は、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.Åj および MSJ (Magic Software Japan K.K.Åj は、いかなる責任も負いません。

本マニュアル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。

本マニュアル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、本マニュアルのいかなる部分も、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 の製品に対するコンパチビリティ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。

本マニュアルにおいて、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限り、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。

Magic は Magic Software Enterprises Ltd. のイスラエルその他の国での商標または登録商標です。

Magic eDeveloper、Magic Client および Magic Application Server は Magic Software Japan K.K. の商標です。

Pervasive.SQL は Pervasive Software, Inc. の商標です。

Microsoft および FrontPage は、Microsoft Corporation の登録商標です。また、Windows、WindowsNT および ActiveX は Microsoft Corporation の商標です。

Oracle は Oracle Corporation の登録商標です。

一般に、会社名、製品名は各社の商標または登録商標です。

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警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。

Version 9.01 第一版

2001年6月25日

Copyright 2001 Magic Software Enterprises Ltd.and Magic Software Japan K.K. All rights reserved.

目次

1 新機能の概要

ブラウザ形式アプリケーション	1
Magic コンポーネント	1
イベント駆動エンジン	1
データ管理	2
エラー処理	2
マルチリクエスト処理	2
生産性の向上	3
開発モードの機能強化	3
外部 Web オーサリングツールの利用	3

2 ブラウザ形式アプリケーション

ブラウザクライアント	5
サーバ側の動作	5
コンテキスト管理	6
ブラウザ形式アプリケーション機能を拡張する関数	7
CALLJS	7
CALLOBJ	7

3 Magic コンポーネント

Magic コンポーネントの統合	9
コンポーネントとは	9
コンポーネントインタフェースの作成	9

4 イベント駆動エンジン

イベント用語	11
イベントタイプ	11
対話型タスクのイベント処理	12
バッチタスクのイベント処理	12
開発時のイベント定義	12
実行時のイベント駆動	13

5 データ管理の改善

遅延トランザクション	15
トランザクションモード	15
トランザクションキャッシュの動作	16
遅延モードで発生するエラー処理	17
更新と削除	17
数値項目の更新	17
DM ステートメントの発行順序	17
リソースロック	18
Lock	18
Unlock	18
リソースロックファイル	18
SQL での範囲指定	18

6 エラー処理

エラー処理の仕組み	21
エラー発生時の処理方式	22
アボート	22
復旧	22
エラーハンドラの作成	22
エラーハンドラの記述	22
実行時のエラー処理	23
レコードポジションに基づくタスクの範囲 / 位置付	23

7 マルチスレッドエンジン

動作環境の設定	25
最大並行リクエスト数	25
現在のアプリケーション	25
INIPUT 関数	25
セキュリティ	25
TCP/IP ポート	25
ブローカと汎用メッセージレイヤ	26
メインプログラム	26
トランザクション	26
ロック	26
メモリテーブル	26
リソースの共有	26
実行時の関数	26

8 GUIの拡張

テーブルコントロール	27
テーブル内でのマルチマーキング	27
マルチマーキングイベントハンドラ	28
マルチマーキング関数	28
テーブルコントロールのサイズ変更	28
ナビゲーションインタフェース	30
リポジトリ	31
タスク	32
クロスリファレンス	32
ブックマーク	32
MVCS	33

9 生産性の向上

クロスリファレンス	35
クロスリファレンスがサポートされるオブジェクト	35
クロスリファレンスの表示	35
メインプログラム	36
メインプログラムの実行モード動作	36
メインプログラムの開発モード動作	36
メインプログラムとコンポーネント	36
メインプログラムでの関数	36
データコントロール	37
コンボボックスとリストボックスの特性	37
モデル	38
モデルリポジトリ	38
モデルを使用した作業	39
Magic フラットファイルでのアプリケーション実行	40
Magic フラットファイルの指定	40
Magic アプリケーションを MFF として保存	40
タスク戻り値	41

10 開発機能の拡張

パラメータ整合	43
セレクトコマンドとパラメータ	43
パラメータテーブル	44
パラメーター一覧	44
パラメータを定義した場合のプログラム呼出	44

パラメータを定義しない場合のプログラム呼出	44
If-Else ブロック	44
コメント	44
参照整合性	46
テーブルリポジトリ	46
APG.....	47
リンク条件	47
開発モード	47
ユーザインタフェース.....	48

新しい Magic eDeveloper の世界へようこそ！ Magic eDeveloper では、拡張されたブラウザ形式の実行パラダイムにより、統合されたエンタープライズレベルのビジネスアプリケーションを効率よく開発するために、多くの機能が新しく追加されています。Magic eDeveloper は、業界をリードするインターネット Web ブラウザ、Web サーバ、アプリケーションサーバ、言語およびデータベースなどをオープンに、かつ多岐にわたってサポートします。

本書では、Magic eDeveloper に新しく追加された機能について解説します。

ブラウザ形式アプリケーション

クライアントのユーザインタフェースとして Web ブラウザを使用することを前提とした、「ブラウザ形式アプリケーション」を効率よく作成することができます。ブラウザ形式の実行パラダイムでは、イベントの検出とそれに対応する処理などのように、関連するデータを含む HTML コントロールを制御するための実行ロジックをクライアント側で提供します。クライアントとサーバとの間の通信処理は、必要最小限に押さえられています。ブラウザ形式の実行パラダイムにおけるサーバ側には、以下の要素が含まれています。

- Web プロセスを実行するための新しいプログラムタイプ
- メインプログラムやイベントドリブン等の新しい概念
- 複数プログラムの並行動作
- タスク実行中のクライアント情報（コンテキスト）の保持

Magic コンポーネント

アプリケーション内のオブジェクトをコンポーネントとして定義できます。コンポーネントをエクスポートすることで、Magic アプリケーション間でリソースを共有する事ができるようになります。

コンポーネントビルダを利用することで、アプリケーション開発時のリポジトリに表示されているオブジェクトからコンポーネントインタフェースファイルを作成することができます。

今までのバージョンでは、アプリケーションオブジェクトを共有することはできませんでした。

イベント駆動エンジン

いろいろな種類のイベントに対応できるようイベントハンドラを定義する仕

第1章 新機能の概要

組みを追加しました。これを利用することで、アプリケーションのタスクフローから受ける制約がなくなります。

データ管理

データ管理機能により、物理データベースに格納されるデータを、より組織的に取り扱うことができます。具体的には次のようなデータ管理機能が新しくサポートされました。

- 遅延トランザクション
- RDBMS の参照整合性制約
- リンク条件
- Magic SQL Where 句

エラー処理

アプリケーション実行時に、エラーハンドラによってデフォルトのエラー処理を置き換えることができます。開発者は、実行時に発生するいろいろなエラーに対して、個別に応答ハンドラを定義することができます。このために、Magic が取得できるエラーの一覧と、使用するそれぞれの DBMS に特有なエラーなど、その他のエラー処理を行うことができる仕組みが提供されます。

マルチリクエスト処理

バックグラウンドのアプリケーションサーバエンジンは、マルチスレッドで動作し、同時に複数のリクエストを処理できるようになりました。各スレッドは異なるコンテキストにより処理を実行するため、相互に影響を及ぼし合うことはありません。

オンラインのアプリケーションサーバエンジンは、開発モード、実行モードともに、今までどおりシングルスレッドで動作します。

第1章 - 新機能の概要

GUIの拡張

ナビゲータや、拡張されたテーブルコントロール、メニュー項目イメージなどにより、動的で使いやすいインターフェースになりました。

生産性の向上

Magic eDeveloper では、生産性向上のため、次のような機能を改良、追加しています。

- クロスリファレンス
- メインプログラム
- データコントロール
- モデル
- Magic フラットファイル (MFF)

開発モードの機能強化

開発モードでは、新たに次のような機能が利用できるようになりました。

- パラメータ項目
- If-Else ブロック
- コメント
- タスクの戻り値

外部 Web オーサリングツールの利用

HTML テンプレートファイルを編集するために、外部のオーサリングツールを利用できます。[設定/動作環境] ダイアログの [アプリケーションサーバ] タブにある [Web オーサリングツール] 欄にて、使用する Web オーサリングツールを指定します。

第1章 新機能の概要

[このページは意図的に空白になっています。]

ブラウザ形式アプリケーション 2

Magic eDeveloper は、ブラウザ形式のアプリケーションを作成するための、総合的なソリューションを提供します。

この章では、次のトピックについて解説します。

- ブラウザクライアント
- サーバ側の動作
- ブラウザ形式アプリケーションの開発機能
- ブラウザ形式アプリケーション機能を拡張する関数

ブラウザクライアント

ブラウザは非常に軽いクライアントであり、クライアント側での保守作業がほとんど必要ありません。

Magic eDeveloper が提供するブラウザクライアントでは、以下のことを実行します。

- Web ページ上で発生したイベントを検出し、それに対応する処理を実行します。
- 再計算を行い、データやその表示インタフェースを更新します。
- ローカルで実行することのできる Magic の関数を処理します。例えば、数値型、文字列、日付、時刻のデータ操作関数を実行することができます。
- データビューの表示を更新します。
- オンラインの Magic クライアントと同様、各項目での入力データの書式チェックをサポートします。
- ローカルキャッシュにデータを保持するため、データが表示領域より大きい場合でも、スクロールにより、ローカルな処理として新しいデータを表示します。これにより、サーバとクライアントの間のやり取りの回数を少なくできます。

これらの新しいクライアント拡張機能は、Magic の開発者にとって分かりやすいものであり、ブラウザクライアントのプログラミング作業をたいへん容易にしています。

サーバ側の動作

サーバ側は、ブラウザ形式のアプリケーションを構成するコンポーネントのなかで最も重要です。サーバでは以下のことを実行します。

- リモートクライアントを識別します。サーバは、各クライアントがオーブ

第2章 ブラウザ形式アプリケーション

ンした論理タスクのタスクコンテキストを個別に保持します。

- タスクを実行しているクライアントの正確な位置やタスクのトランザクション内で操作されるデータについての内部記述として、コンテキストを確定します。
- クライアントから送られてきたデータ処理要求を解読し、該当するデータベースに反映させます。
- 新たにリンクされたレコードが発生した場合、クライアント上のレコードを再リンクし、反映させます。
- ブラウザクライアントの性質上、ローカルでは実行できない関数を実行します。

サーバがブラウザクライアントを作成するとき、ローカルで処理を実行するタイミングや、また関数を使用したロジックを実行するためにサーバをいつ呼べばいいのかといった " 指示 " をクライアントに対して行ないます。

コンテキスト管理

コンテキストマネージャは、全てのリクエストを受け取り、それを振り分ける機能を持っており、有効なコンテキストの維持管理やユーザからのリクエスト処理の責任を持っています。

ブラウザ形式の実行パラダイムに対するコンテキスト管理メカニズムによって、ブラウザクライアントとの間でセッションが有効になり、やり取りが行われている間、プログラムが有効な状態に維持されます。コンテキストの中には、実行エンジン、タスクツリー、データベースのカーソル、メモリーテーブル、入出力ファイルなどの状態情報が含まれています。

各コンテキストは、個別の ID 番号を持っています。この ID 番号により、各クライアントに対応するコンテキストが認識されます。

コンテキストは、プログラムが終了するか、[コンテキスト非稼動タイムアウト] が経過するまで有効です。[コンテキスト非稼動タイムアウト] は、[設定 / 動作環境] ダイアログにて秒単位で指定します。

第 2 章 - ブラウザ形式アプリケーション

ブラウザ形式アプリケーションの開発機能

ブラウザ形式アプリケーションに関する開発機能について解説します。

タスクのタイプとして、[O= オンライン] と [B= バッチ] に加え、新たに [R= ブラウザ] が追加されました。このタスクタイプを指定することにより、実行時にオンラインのブラウザクライアントとなるプログラムを作成することができます。

ブラウザのインタフェースは HTML によって記述されますが、開発者の便宜のため、日頃から使用しているサードパーティ製の Web オーサリングツールを Magic から容易に利用できる機能が提供されます。このとき、Magic は全ての Web ページ内エレメントとスムーズに連携します。

外部のオーサリングツールを利用して HTML を記述する場合、ユーザインタフェースは Magic の管理対象ではなくなりますが、Magic eDeveloper からは、Web ページ上の各コントロール特性に式を設定することにより、データ駆動コントロールとして取り扱うことができるようになります。

ブラウザ形式アプリケーション機能を拡張する関数

CALLJS

Web ページ内にある Java スクリプト関数を呼び出します。

CALLOBJ

Web ページ内にある ActiveX のメソッドや Java アプレットオブジェクトを呼び出します。

第2章 ブラウザ形式アプリケーション

[このページは意図的に空白になっています。]

Magic コンポーネント

3

Magic eDeveloper では、アプリケーション内のオブジェクトをコンポーネントとして定義できます。コンポーネントとしてエクスポートすることで、アプリケーション間でのリソース共有が出来るようになります。

この章では、次のトピックについて解説します。

- Magic コンポーネントの統合
- コンポーネントとは
- コンポーネントインタフェースの作成

Magic コンポーネントの統合

Magic eDeveloper では、作成したアプリケーションを複数のコンポーネントに分割することができます。そしてそれぞれのコンポーネントには、アプリケーションで定義したオブジェクトを内包させることができます。

コンポーネントとは

コンポーネントは、開発モードにおいて作業対象となっている開発中の Magic アプリケーションに対して、付加的に使用されるもう一つの Magic アプリケーションと考えることができます。

コンポーネントは、MCI (Magic Component Interface) ファイルのインタフェース定義内容を読み込むことで追加することができます。このファイルは拡張子が mci のテキストファイルで、開発者に対して必要な情報、すなわち付加されるアプリケーションにおいて参照し利用できるオブジェクトについて記述されています。

MCI によって公開されるオブジェクトは、モデル、テーブル、プログラム、ヘルプ、権利、メインプログラムのイベント、環境設定のサブセット、論理名、データベース定義です。

コンポーネントのインタフェースが読み込まれると、付加されたアプリケーション内の公開されているオブジェクトを、開発中のアプリケーション自体内で定義しているものと同じように参照することができるようになります。

コンポーネントインタフェースの作成

MCI ファイルは単純なテキストファイルですが、Magic eDeveloper ではこのファイルの作成用ユーティリティとして、コンポーネントビルダを提供しています。このユーティリティを利用すれば、他のアプリケーションに利用させるコンポーネントオブジェクトを簡単に定義することができます。

第3章 Magic コンポーネント

[このページは意図的に空白になっています。]

イベント駆動エンジン

4

Magic eDeveloper では、タスク実行時に発生する暗黙的または明示的なイベントについて、その対応処理を開発者が定義することができます。

この章では、次のトピックについて解説します。

- イベント用語
- イベントタイプ
- オンラインタスクのイベント処理
- バッチタスクのイベント処理
- 開発時のイベント駆動
- 実行時のイベント駆動

イベント用語

イベント駆動エンジンで用いられる用語は次のとおりです。

- **イベント**.....意味のある事象の発生に対する論理的な定義です。イベントの発生により、それに対応するイベントハンドラが起動され、一連の処理が実行されます。
- **トリガ**.....イベントを発生させる実際のアクションを定義したものです。イベントは他のユーザ定義イベントのトリガとして割り当てることができます。つまりトリガとなるイベントが発生すると、同時にユーザ定義イベントも発生することになります。
- **イベントハンドラ**.....イベントが発生したときに実行される、一連の処理を定義したものです。

イベントタイプ

Magic のイベントは、「事前定義イベント」と「ユーザ定義イベント」という2つのグループに分類されます。

事前定義イベントは、次のカテゴリに分類されます。

- **内部 Magic イベント**.....これには、次のものがあります。
 - Magic 実行処理が定義するイベント
 - Magic エンジンにハードコードされたハンドラを含むイベント。例：前処理、メイン、後処理など。

第4章 イベント駆動エンジン

- マウスボタンをトリガにしたイベント。マウスボタンのトリガには、`onclick, ondblclick, onmouseover, onmouseout` があります。
 - システムイベント.....ユーザがさまざまなキーを組み合わせてキーボードから入力することにより発生します。
- ユーザ定義イベントは、次のカテゴリに分類されます。
- タイマイベント.....事前に定義された間隔で発生するイベント。
 - 式評価イベント.....式が真と評価された場合に発生するイベント。

対話型タスクのイベント処理

オンラインタスクやブラウザタスクのような対話型タスクでは、コントロールレベルと呼ばれる新しい処理レベルがタスクフローに追加されました。コントロールレベルには、あらかじめ2つ（コントロール前処理、コントロール後処理）のイベントハンドラが定義されていて、値の設定、入力の確認、表示の制御などに利用することができます。

Magic は、コントロールからの入力中にイベントの発生を検出することができます。イベントが発生したときは、現在のタスクのスコープ内でイベントを処理できるのか、またはタスクの変更が要求されているのかを判断します。

バッチタスクのイベント処理

バッチタスクの実行処理は、オンラインタスクのようなキー操作が入る処理とは異なります。オンラインタスクの場合、実行エンジンはキー入力を待ちますが、バッチタスクではキー入力を待たずにレコードを処理します。

バッチタスクでは、処理中にイベントの発生をチェックすることができます。すなわちタイマ時間経過や、処理されたレコード数によってイベント発生をチェックするよう、バッチタスクを定義することができます。

タイマ時間は、[動作環境] ダイアログの [バッチイベント間隔 (ミリ秒)] というパラメータで指定します。

レコード数によるイベントは、[タスク制御] ダイアログの [レコードイベント間隔] というパラメータに、数値型の値を返す定義式を指定します。

開発時のイベント定義

[イベント] テーブルにてイベントを定義することができます。イベントは特定のトリガに対応して設定されます。定義済みのイベントまたは、直接トリガに対してイベントハンドラを定義できます。

ユーザ定義イベント

開発者はタスク内に独自のイベントを定義することができます。[イベント] テーブルには、現在タスクの親（上位）タスクで定義されたすべてのユーザ定義イベントが表示され、現在タスク内で定義するユーザイベントについてのみ、変更を行うことができます。

ユーザ定義イベントは、それを定義したタスクとその子（下位）タスクから参照することができます。メインプログラムで定義されたイベントは、アプリケーションのすべてのプログラムから参照されるため、グローバルイベン

第4章 - イベント駆動エンジン

トになります。

ユーザ定義イベントは、そのトリガとともに定義することができます。トリガは補助的なイベントであり、それが発生したときに、ユーザ定義イベントを暗黙的に起動し発生させるものです。

ハンドラテーブル

[ハンドラ]テーブルでイベントに対応するハンドラを定義できます。[ハンドラ]テーブルはタスク内で見ることができ、ハンドラの処理フローやパラメータとしての変数項目、などを定義することができます。

イベント実行コマンド

[イベント実行]という新しいコマンドを使用して、Magic エンジンの動作フローの中でイベントを発生させることができます。このコマンドは、システムイベントと同じように処理されます。

実行時のイベント駆動

ハンドラ検索機能の仕組み

新しいタスクが読み込まれると、実行エンジンはイベントハンドラを探します。各ハンドラは、トリガとなるイベントやタスクツリー上の位置に従って検索順序がメモリ上に構築されます。タスクが終了すると、そのタスクで定義されているハンドラはメモリ上から削除されます。

イベントが発生すると、検索順序に従ってイベントハンドラを探します。ハンドラの検索は、最後に登録されたものから最初に登録されたものに向かう順番に行われます。

あるタスクレベルでは処理できないイベントと判断されると、実行エンジンはタスクツリーでのより上位タスクのイベントハンドラを検索対象としてゆきます。

伝播

あるイベントをより高いレベルで定義されたイベントハンドラに渡してゆくことができます。あるイベントハンドラが実行された後、[伝播]が「Yes」に指定されているときは、ディスパッチャは次のレベルのイベントハンドラを探して処理しようとします。[伝播]が「No」のときは、イベントはそのイベントハンドラの処理後ブロックされ、次のハンドラには伝播されません。

コンポーネントイベント

イベントをコンポーネント化することで、呼び出し元アプリケーションが処理できるイベントを提供することができます。

また、コンポーネントアプリケーションのメインプログラムにイベントハンドラを定義することで、呼び出し元アプリケーションで発生したイベントに対するハンドラを実行することができます。

第4章 イベント駆動エンジン

[このページは意図的に空白になっています。]

Magic eDeveloper では、データの完全性、整合性を維持するための機能を強化する一方、プログラミング作業をよりシンプルにする開発環境を提供します。

この章では、次のトピックについて解説します。

- 遅延トランザクション
- DM ステートメントの発行順序
- リソースロック
- SQL での範囲指定

遅延トランザクション

今までのバージョンでは、レコード後処理の後に DM ステートメント (insert/update/delete など) をデータベースシステムに対して発行していました。物理トランザクションと呼ばれるこの処理方法では、これらの DM ステートメントが発行される毎に、物理データベースの内容を更新します。アプリケーション実行時に、データビューのレコード内容の変更を反映するためにこの処理を行っていますが、(レコード毎に行うため) 時間のかかるものでした。

遅延トランザクションは、データビューにあるレコードの変更をコミットするまで、DM ステートメントの発行を遅らせます。dbMAGIC は、発行する DM ステートメントの内容をキャッシュに貯めておきます。データベースへの変更をコミットすることにより、全ての変更処理が整合性を保ちながら実行されます。

トランザクションモード

[トランザクションモード] の設定欄が、[タスク特性] ダイアログに追加されました。トランザクションモードとして、次の 4 つのうちから選択します。

- **遅延**.....DM ステートメントがキャッシュに保存されます。タスクの処理中は、DM ステートメントは発行されません。ステートメントはコミット時にのみ発行されます。キャッシュに保存されていた全ての DM ステートメントは一度に発行され、その後トランザクションがクローズされます。
- **ネスト遅延**.....他の遅延トランザクションタスクからコールされたタスクの場合に、親タスクの遅延トランザクション内にネストされた新しい遅延トランザクションをオープンします。それ以外は、"遅延" と同じです。
- **物理**.....レコード後処理の後に DM ステートメントを発行します。(この方式は、今までのバージョンと同じです。)
- **親と同じ**.....親と同じトランザクションモードでタスクが実行されます。

第5章 データ管理の改善

親が物理トランザクションの場合、物理トランザクション。親が遅延トランザクションの場合、遅延トランザクションとして処理されます。

ブレイクレベルのトランザクションパラメータは、タスク特性のグループレベルで設定するようになりました。

トランザクション処理

トランザクションが " 遅延 " に設定された場合、3つの異なるタスクレベルで次のような処理が行われます。

- **レコード**.....レコードレベルでの全ての更新内容が、1つにまとめられます。レコードの更新時にトランザクションがオープンされ、変更内容が適用されて、トランザクションがコミットされます。
- **タスク**.....タスクレベルでの全ての更新内容が、1つにまとめられます。タスク後処理の後にトランザクションがオープンされ、変更内容が適用されて、トランザクションがコミットされます。
- **グループ**.....更新内容はタスクレベルでまとめられ、特定の基準とする項目の値が変わるときにトランザクションがオープンされてコミットされます。グループレベルのトランザクションを使用するときは、グループ項目を定義しなければなりません。これはバッチタスクでのみ設定できます。

ロック方式

遅延トランザクションでは、次のロック方式を指定できます。

- N= なし
- O= 入力時

SQL Where 句

SQL Where 句を遅延トランザクションモードのタスクで動作させるため、定義方法が変更されています。

埋め込み SQL

埋め込み SQL のタスクは、物理トランザクションモードのみサポートされています。

トランザクションキャッシュの動作

- トランザクションのキャッシュには、遅延トランザクション中に変更された全てのデータを保存しています。
- 各遅延トランザクションは、個別にトランザクションキャッシュを持っています。
- 遅延トランザクション内にある全てのタスクは、アクセスする全てのファイルの全てのデータをキャッシュするようにします。
- 各テーブルは、自身のテーブルキャッシュを持っています。Magic のテーブルリポジトリへの登録は、通常一つのテーブルに対応します。二つの異なる登録が同じテーブルに対応する場合でもキャッシュは別々に管理され、共有しません。

遅延モードで発生するエラー処理

遅延トランザクションモードで動作中のときは、DM ステートメントが対象となる物理データベースシステムで実際に実行されるときにのみエラーの検出が行われます。そのエラーやテーブルに対して定義されたエラーハンドラは、そのエラーの発生により起動されます。エラーハンドラが起動されると、ハンドラ用に定義されたファイルやテーブルだけが表示されます。

エラーハンドラが動作すると、次のものについて調べることができます。

- エラー発生時にキャッシュされていたレコードの値
- エラーコード
- エラーメッセージ

更新と削除

発行される update や delete の各コマンドに対する Where 句の作成方法を Magic に指定することができます。

- **位置のみ**.....位置項目のみ含む
- **位置と更新項目**.....位置項目と更新される項目の元の値を含む
- **位置と選択項目**.....位置項目と選択された項目の元の値を含む

[更新レコードの識別] という新しい設定項目が [テーブル特性] に追加されました。この設定として、上記の値のいずれかを選択することができます。この設定は、遅延トランザクションタスクでのみ有効です。この設定は、タスクの DB テーブルにも追加されました。

数値項目の更新

今までのバージョンでは、数値を直接指定して更新していました。例えば、FLD1 という項目を X という値に更新する場合、(FLD1=X) の形式でコマンドを発行します。

Magic eDeveloper では、差分更新もできるようにしました。例えば、上の例では、更新前の FLD1 と X の差分 x を Magic 内部で求めた上で、(FLD1=FLD1+x) の形式でコマンドを発行します。

この [更新形式] という特性が、[カラム特性] に追加されました。選択できる方法は次の二つです。

- **A= 値更新**.....旧バージョンと同じ方法で値を更新します :(FLD1=X)
- **D= 差分更新**.....差分の値で更新します :(FLD1=FLD1+x)

この特性は、SQL テーブルで通常の記憶形式を持った数値項目に対してのみ有効です。ISAM ファイルではサポートされません。

この更新方法の設定は、実項目に対するセレクトコマンドにも追加されます。「A= 値更新」や「D= 差分更新」という値に加えて、「T= テーブル特性に依存」という値が指定できます。この場合は、[カラム特性] の設定内容と同じになります。

DM ステートメントの発行順序

第5章 データ管理の改善

遅延トランザクション中の DM ステートメント発行順序を指定することができます。今までのバージョンでは、DM ステートメントの発行順はレコード後処理の実行順序に依存していました。例えば、親タスクでの更新が子タスクの起動前に行われていても、起動された子タスクでの更新に対する DM ステートメントの方が、親タスクでの更新に対する DM ステートメントよりも先に発行されました。

Magic eDeveloper では、コールコマンドの [コール] 特性ダイアログに [同期] という設定が追加されました。

この設定には、次の値を設定します。

- No.....旧バージョンと同じ（コール先の実行コマンドを先に発行）
- Yes.....コール元の DM コマンドを先に発行
- 式.....Yes または No となる論理式の結果にしたがって発行順序が決定

リソースロック

リソースロックとは、ある瞬間にはただ一人のユーザのみによって占有される仮想的な要素（リソース）を作成して、ロックを実現することです。それぞれのリソース名はユニークにしなければなりません。

リソースロックを制御する二つの関数があります。

Lock

LOCK 関数は、通常項目更新の式で設定し、引数として（別のユーザによりロックされた）リソースを待つ時間の長さを設定できます。ロックが成功したときは戻り値として「0」を返します。リソースがすでに自身のタスクでロックされているときは「1」、別のユーザがロックした状態でタイムアウトを越えた場合は「2」を返してロックは失敗します。

Unlock

UNLOCK 関数は、リソースのロックを解放するために式で設定します。アンロックが成功したときは戻り値として「0」を返します。リソースがすでにアンロックされているか、他のユーザによってロックされているとき、アンロックは失敗します。

リソースロックファイル

[動作環境] ダイアログの [マルチユーザ] タブに、[リソースロックファイル] が追加されました。

Magic.ini では、"ResourceLockFilePath" という名前のリソースファイルのパス名が新しく追加されました。

SQL での範囲指定

今までの SQL Where 句による範囲指定は、遅延トランザクション内のタスクでは許可されていません。遅延トランザクション内で SQL Where 句による範

第5章 - データ管理の改善

囲指定と同等のことは実現させるには、開発時に範囲定義の方法を変更しておく必要があります。

今までの範囲指定の方法には次のものがあります。

- **セレクトコマンドの範囲欄**..... セレクトコマンドで範囲を定義します。この範囲指定は、タスクがオープンされる時実行されます。また、ビュー再表示アクションの実行によってのみ更新されます。
- **SQL Where 句**..... SQL タスクのみ有効です。SQL Where 句の範囲は、タスク実行時に Magic が作成する SQL コマンドに追加される SQL Where 句に含まれます。
- **(タスク) 範囲式**..... タスク制御の設定で、ここには式を指定します。この式は、レコードを読み込む毎に評価されます。タスク範囲を超えたレコードは "False" となり、読み込まれません。

後の2つの範囲指定は、タスク環境メニューの中の新しい [範囲 / 位置付] ダイアログのタブセクションとして整理されました。

Magic eDeveloper では "Magic SQL Where 句" と呼ぶ4番目の範囲指定方法が追加されました。これは、Ver8 での SQL Where 句とほぼ同等の機能ですが、フリーフォーマットのテキストを使用する既存の SQL Where 句とは異なり、式で設定します。つまり Magic SQL Where 句には、Magic エンジンが SQL Where 句として変換できる式を記述します。

Magic SQL Where 句は、次のように実行されます。

- 実行時、Magic SQL Where 句は、AND を使用して他の範囲条件に追加されます。
- 遅延トランザクション内では、SQL Where 句は無効です。Magic は、残りの3つの範囲指定方法を使用します。
- 整理すると、既存の範囲処理方法は次のようになります。
 - **SQL Where 句**..... [SQL Where 句] の [DB SQL]
 - **タスクの範囲指定**..... [範囲 / 位置付] の [範囲式]
 - **新しい "Magic SQL Where 句"**..... [SQL Where 句] の [Magic SQL]
- 新しい [範囲 / 位置付] ダイアログには、実行時に生成される Where 句がまとめて表示されます。セレクトコマンドの [範囲] 欄、Magic SQL Where 句、[範囲 / 位置付] の [範囲式]、そして物理トランザクションタスクの場合には、DB SQL Where 句の内容も表示されます。

注意： DB SQL Where 句 と Magic SQL Where 句 は、タスク前処理の前に一度だけ処理されます。

第5章 データ管理の改善

[このページは意図的に空白になっています。]

Magic eDeveloper のエラー処理機能の特徴は、アプリケーション実行時に発生するいろいろなエラーに対応するデフォルトの動作を、開発者独自のものに変更できるということです。

この章では、次のトピックについて解説します。

- エラー処理の仕組み
- エラー発生時の処理方式
- エラーハンドラの作成
- 実行時のエラー処理
- レコードの位置に基づくタスクの範囲 / 位置付

エラー処理の仕組み

エラー処理の仕組みによって、アプリケーション実行中に発生したエラーに対応するデフォルトの動作を変更することができます。

エラーが発生したときには、2つのレベルで動作を制御できます。最初のレベルは、今までのバージョンでのエラー発生時の処理内容とほぼ同様ですが、より動作の安定性を高めたものになっています。このレベルでのエラー対応動作の方式は2つあり、タスク特性ダイアログにてどちらかを指定します。この詳細については以下に記述します。

2つ目はより細やかな対応のできるエラー処理レベルで、エラーに対応する処理内容を Magic のプログラムとして記述し、エラーハンドラとして使用するものです。Magic エンジンは検出できる既知のエラーや予想されるエラーの一覧を表示し、使用する DBMS 特有のエラーコードに対応する処理を可能にします。このレベルでは、エラーハンドラ処理の後に実行する、Magic エンジン側の対応動作を指定することができます。

この章では、データベースに関連したエラー処理について説明します。

エラー発生時の処理方式

エラー発生時の処理方式としてアボートと復旧の 2 つがあり、タスクレベルの設定としてタスク特性にて指定します。この処理方式の設定を注意深く行うことにより、多くの場合にはデフォルトのエラー処理を変更したり、特別なハンドラを書く必要はありません。もちろんデフォルトの方法ではエラーに対応しきれない場合には、エラーハンドラを書くことで、デフォルトの設定を変更できます。

アボート

エラーが発生した場合、Magic は現在のトランザクションをロールバックします。物理トランザクションおよび遅延トランザクションのいずれの場合にも、読み込んだデータをクリアし、エラーが発生したタスクをアボートします。ただしこの方式は、全てのエラーに対して適用されるものではなく、ロックエラーや不正ログインのような、エンドユーザの操作により回復できるエラーは例外となります。

復旧

可能な限りタスクを続行し、読み込んだデータを保持してエンドユーザがそのエラーを復旧できるようにします。つまりこの場合は、エラーが発生した後でも、エンドユーザがそのアプリケーションの作業を継続できることを意味します。また、ある種のエラー、たとえばロックエラーや最大接続数を越えた場合のエラーなどに対しては、エンドユーザの介入なしに、Magic は自動的に処理を繰り返します。

エラーハンドラの作成

エラーハンドラは [タスク定義] とその [処理テーブル] にて作成します。これはイベントハンドラの場合と同様です。

エラーハンドラの記述

エラーハンドラとして、次の記述項目があります。

- レベル
- イベントタイプ / イベント
- スコープ
- 有効
- エンジン対応動作指定

レベル

[レベル] カラムで「H= ハンドラ」を選択します。

イベントタイプ / イベント

[イベント] カラムでズームし、イベントタイプとして「R= エラー」を設定するとともに、実際のエラーとなるイベントを [エラー一覧] から選択します。

スコープ

[スコープ] カラムでは、以下のオプションから選択します。このハンドラが特定のタスクのみで実行するものか、タスクのサブツリー全体で実行するかを指定します。

- サブツリー.....デフォルト：タスクのサブツリー全体で実行します。
- タスク.....そのタスクで検出されたもののみ実行され、サブツリーで発生したエラーは無視します。
- グローバル..... コンポーネントのメインプログラムでのみ有効です。

有効

ハンドラが有効かどうかを指定します。この特性は定義式で指定できます。「False」と評価された場合は、ハンドラはエラーを受け取りません。

対応動作

[詳細] カラムの [対応 :] 欄で、後処理を指定します。Magic エンジンに用意されている処理のなかから、エラーハンドラ実行後に行う動作を指定します。

また、[メッセージ :] 欄で「Yes/No」を設定することにより、Magic がデフォルトのメッセージを表示するか否かを指定します。

実行時のエラー処理

エラー状態になったとき、エラー処理機能は同じタスク内で定義されたエラーハンドラを、エラー名を元に探します。エラーハンドラが見つかったら、そのハンドラによって定義された処理を実行します。そして、ハンドラのエンジン側対応動作で指定された内容にしたがう処理を引き続いて実行します。エラーハンドラに対する検索は、イベントハンドラと同じように行われます。ハンドラがタスクに存在しない場合は、より上位のタスクに対して該当するハンドラの検索を行います。

検索の結果、対応する特定ハンドラがどこにも見つからなかった場合、「なんでも」に対するエラーハンドラが定義されていれば、それが実行されます。

レコードポジションに基づくタスクの範囲 / 位置付

エラー処理の重要な仕様として、エラーが発生したポジション情報に基づいて、そのエラーレコードやレコードの範囲を表示できます。そのためには次のような操作を行ないます。

1. CURRPOSITION 関数によって現在のポジションを取得
2. ERRPOSITION 関数によってエラー発生ポジションを取得

第 6 章 エラー処理

3. 得られたポジション情報にしたがってタスクを位置付ける

マルチスレッドエンジン

7

マルチスレッド化により、バックグラウンドモードで動作するエンジン1つで、複数のリクエストを同時に処理できるようになります。マルチスレッドアーキテクチャによってリソースの節約を図るとともに、高いパフォーマンスを提供します。

この章では、次のトピックについて解説します。

- 動作環境の設定
- 実行時の関数

動作環境の設定

最大並行リクエスト数

「最大並行リクエスト数」の設定は、[動作環境] ダイアログの [アプリケーションサーバ] タブにあり、アプリケーションサーバが作成するスレッドの上限の数を指定できます。実際に並行動作するスレッド数は、ここに設定した値とサーバの能力により物理的に決まる数や許諾されているライセンスによって制限される数のうち最も小さい値となります。

MAGIC.INI 及び、コマンドラインでのパラメータは MaxConcurrentRequests です。

現在のアプリケーション

1つのプロセスで動作する全てのスレッドは、常に同じアプリケーションが動作します。そのアプリケーションをクローズするためには、全てのスレッドをクローズしなければなりません。

INIPUT 関数

MAGIC.INI ファイルは、別々のメモリ領域内にあるそれぞれの実行コンテキストに格納されます。強制書込みを指定した時だけ、INIPUT 関数にてその実行時コンテキスト中の MAGIC.INI で更新します。

INIPUT (<INI 項目>,<強制書込>) (強制書込デフォルトは No)

セキュリティ

実行時に関数を使用して、各スレッドに権利を設定したり解除したりすることができます。

TCP/IP ポート

アプリケーションサーバ各プロセスは1つのTCP/IPポートを使用します。

ブローカと汎用メッセージレイヤ

Magic eDeveloper と前バージョンとで、相互運用することはできません。

メインプログラム

各コンテキストは、メインプログラムを個別にコピーします。

トランザクション

各スレッドは独立しており、別々のプロセスとして動作します。

ロック

各スレッドは別々のプロセスとして動作し、アプリケーションサーバプロセスはそれぞれ1つの端末番号を使用します。

メモリテーブル

各コンテキストは、メモリテーブルを個別にオープンします。

リソースの共有

次のリソースは各実行スレッドで共有されます。

- DBMS 接続
- 外部デバイス

実行時の関数

次に示すような関数があります。

- **RQEXE**.....実行エンジンのリストに登録された内容をもとにアプリケーションの実行をブローカに要求します。
- **RQRTTRM**.....アプリケーションサーバを終了させます。
- **RQRTINF**.....実行中のスレッドの数や、ライセンスで許可されたスレッドの最大数、通常使用されているスレッド数を返します。
- **RQRTINF**.....実行可能なスレッド数を返します。
- **DISCSRVR**.....使用しないスレッドを切断します。
- **DBRELOAD**.....実行時に常駐テーブルを読み込み、読み込みに失敗したか成功したかを戻り値として返します。

Magic eDeveloper の外観が新しくなりました。ワークスペースをマルチに切り替えるナビゲータペインやリポジトリテーブルのフォルダ、テーブルのマルチマーキングなどにより、ダイナミックな使いやすいユーザインタフェースになりました。

この章では、次のトピックについて解説します。

- テーブルコントロール
- ナビゲーションインタフェース

テーブルコントロール

Magic eDeveloper は、テーブルコントロールの仕様が強化され、開発時や実行時において、マルチマーキング、フォームサイズ変更にもなう再配置、カラムの取扱い（自動ソートやサイズの変更）などが新たにできるようになりました。

テーブル内でのマルチマーキング

マルチマーキングは、Windows アプリケーションでの複数同時選択と同様の機能です。

マルチマーキングの設定

テーブルコントロールでマルチマーキングを有効にするには、[テーブルコントロール特性] の [マルチマーキング] の設定を「Yes」にします。マルチマーキングは、オンラインタスクでのみ設定が有効になります。

マークされた行は、異なる色で表示されます。

マークされたレコードに対し、次のことを実行できます。

- 構文チェック.....開発モード
- コピー.....開発モード
- 削除.....開発モードと実行モード
- イベント.....実行モード（マルチマーキング用ハンドラを作成して独自のロジックを定義）

マーキングカラム

カラムをマーキング出来るようにするには、[カラム特性] の [マーキングカラム] を「Yes」にします。マーキングカラムが有効の場合、カラムのレコード部分が浮き上がって表示されます。カラムをクリックすると、レコードがマークされます。マークされたレコードをクリックすると、マークが取り消されます。

マルチマーキングイベントハンドラ

数のレコードがマークされている状態のときにイベントが発生すると、それぞれのレコードに対してバッチ処理的にハンドラが実行されます。

また、各レコードにおいてレコード前処理が実行され、レコード内容が変更されたり、強制レコード後処理の指定がある場合には、レコード後処理が実行されます。

マークされたレコードに対して、ハンドラでの処理フローが同じように実行されるので、いくつか関数が新しく追加され、複数のレコードがマークされていることや、どのレコードが最初にマークされ、どのレコードが最後にマークされたかが分かるようになっています。

レコードの処理は、データビューでの位置に従って行われ、マークされた順番で行われるわけではありません。

マルチマーキング関数

マルチマーキング関数として次のものがあります。

- **MMCOUNT**..... マークされたレコード行数を返します。「0」よりも大きな値を返したときは、レコードがマークされていることを意味します。
- **MMCURR**..... マークされたレコード行全体のうちで現在行が何番目であるのかを示します。
- **MMSTOP**..... マルチマーキングハンドラを停止させます。

MMCURR 関数が「1」を返したときは、マークされた最初のレコードが処理されていることを意味します。

MMCURR 関数が、MMCOUNT 関数の戻り値と同じ値を返したときは、マークされた最後のレコードが処理されていることを意味します。

テーブルコントロールのサイズ変更

テーブルコントロールを含むフォームの大きさを変更すると、テーブルコントロールもそれに従って大きさが調整されます。

水平方向の場合、テーブルコントロールには [位置とサイズ / 位置] パラメータがあり、テーブルに対する変更パーセントで指定します。[位置とサイズ / 位置] パラメータが 80% と指定された場合、フォームのサイズが 100 ピクセル増えた場合、テーブルコントロールのサイズは 80 ピクセル増えることとなります。

垂直方向の場合、テーブルコントロールは、どのくらいフォームのサイズが変更されたかにより行数が増えたり減ったりします。データがなくなる場合は、空の行を表示します。パークできるテーブルコントロールの行は常に表示されます。

開発モードでのカラムサイズ変更

各カラムはテーブルのサイズ変更にもなってサイズが変わります。カラムには、[位置とサイズ / 位置] パラメータがあります。この設定が「Yes」の場合、カラムは、カラムの総数に比例したサイズ変更をします。例えば、4つのカラムがあり、その [位置とサイズ / 位置] パラメータに「Yes」が設定された場合、テーブル変更量の 25% ずつがサイズに反映されます。

第 8 章 - GUI の拡張

[位置とサイズ/位置]パラメータに「No」が設定された場合、カラムのサイズは変更されません。例えば、80 ピクセル幅のテーブルコントロールを想定します。位置とサイズの特性を 100% に設定した場合、4 つのカラムは 20 ピクセルの幅になります。このとき 2 つのカラムの [位置とサイズ/位置]パラメータが「No」で、残りの 2 つのカラムの [位置とサイズ/位置]パラメータが「Yes」とします。フォームのグリッドが 8 ピクセル増加した場合、2 つのカラムは 20 ピクセル幅で残りの 2 つのカラムは 24 ピクセル幅になります。

すべてのカラムの [位置とサイズ/位置]パラメータに「No」の設定をすると、テーブルコントロールの [位置とサイズ/項目位置]にしかるべき設定がしてあっても、水平方向のテーブルサイズは変更されません。

実行モードでのカラムのサイズ変更

開発者は、カラムのサイズ変更特性を「Yes」にすることで、実行時にテーブルのカラムのサイズを変更することができます。右の区切り線によってカラムのサイズを増やしたり残りのカラムを右に押し出したり出来ます。残りのカラムのサイズは変わりません。このため、カラムのサイズを大きくするとテーブルのサイズも大きくなります。しかし、フォーム上に表示されるテーブルのサイズが変わらず、表示されるカラムが減らされます。テーブルのサイズが変更され、テーブルのカラムが全部表示されなくなった場合、水平スクロールバーがテーブルに表示されます。

ナビゲーションインタフェース

Magic eDeveloper は、開発リポジトリがより使いやすくなり、ユーザーフレンドリなナビゲータペインを利用できるようになりました。アプリケーションをオープンしたときのメインスクリーンは以下のように表示されます。

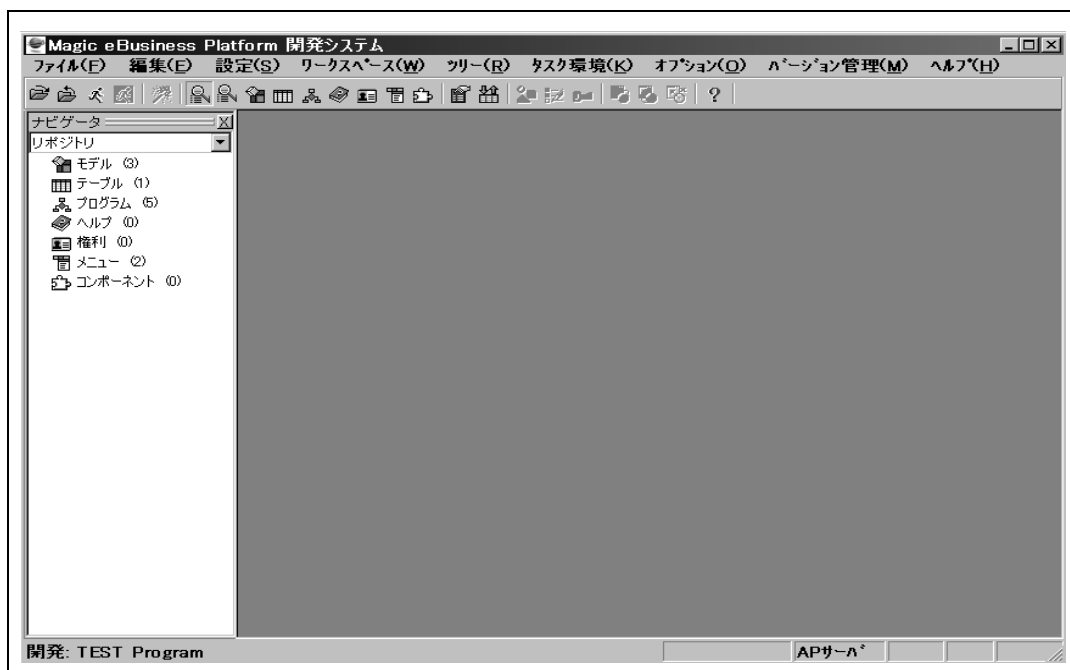


図 8-1 Magic のメイン画面

ナビゲータペインを利用して、アプリケーションの各リポジトリにアクセスすることができます。またナビゲータペインのサイズを小さくして、ワークスペース領域を大きく取ることができます。

ワークスペースには、リポジトリのデータや設定、各特性、アプリケーションのフォームを表示します。画面のフォーカスを切り替えるためのショートカットを定義することもできます。

ナビゲータペインは、デフォルトでは画面の左側に表示されます。また、Window 内の任意の辺に寄せることもできます。あるいは、辺に固定させるのではなく、フローティングさせることもできます。

ナビゲータペインの表示内容は、次の 4 つから選択できます。

- リポジトリ
- タスク
- ブックマーク
- クロスリファレンス
- MVCS (チーム開発時のみ有効)

アプリケーション開発にスペースが必要な場合、ナビゲータペインはいつでも閉じることができます。開発モードから実行モードに切り替わったり、プログラムを起動して実行した場合には、自動的に閉じられます。

リポジトリ

リポジトリを選択したときには、フォルダのツリーが表示されます。メインのフォルダには、モデルやテーブル、プログラムなどの各リポジトリのオブジェクトが表示されます。フォルダツリーは、サブツリーとして1つ下のレベルまで展開され、それぞれのリポジトリでユーザが定義したフォルダを確認することができます。



図 8-2 リポジトリの使用

フォルダツリーには次のような情報が表示されます。

- フォルダイメージの展開と縮小
- フォルダ名
- 登録オブジェクトの数

フォルダは、リポジトリへの登録内容を組織化するためのオブジェクトです。これを利用することにより、ユーザはリポジトリへの登録内容を、範囲によって分類することができます。例えば、あるフォルダにテーブルの10番から19番、別のフォルダに20番から29番を入れるといった使い方です。

各リポジトリには、フォルダを100個まで持つことができます。各フォルダは、リポジトリ内で登録されている順番に対応しています。ユーザがあるフォルダから別のフォルダへ1つの登録内容を移した場合、登録リストの連続性を維持するため、自動的にリポジトリ内の登録項番を付けなおします。

ユーザ定義フォルダではなく、メインのフォルダに入力された項目は、リポジトリ登録リストの先頭に置かれ、リポジトリ全体が表示されるときに見ることができます。

アプリケーションの各リポジトリは、ナビゲータでリポジトリを選択することにより表示されます。ユーザは次の各リポジトリでユーザ定義フォルダを

第 8 章 GUI の拡張

作成することができます。

- 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

各リポジトリの登録リストにあるフォルダカラムの内容を見ることにより、オブジェクトが登録してあるフォルダを確認できます。

タスク

プログラムを開発するとき、ナビゲータペインのタスク表示が有効に利用できます。

ナビゲータペインのタスク表示により、プログラムのタスクツリー構造を確認することができます。

タスクツリーが表示されているとき、次の操作を行うことができます。

- メインタスクやサブタスクのナビゲーションによる概要確認
- プログラム内ツリー構造の修正操作
- タスクのサブツリー切り取りやコピーと貼り付け

クロスリファレンス

ナビゲータペインにはクロスリファレンス表示があり、クロスリファレンスの検索結果を保持しています。それぞれのクロスリファレンスの検索結果は、ツリーフォーマットで表示されます。

表示された検索結果オブジェクトをクリックすることで、ワークスペース側はそのオブジェクトの存在する位置が表示されるよう切り変わり、カーソルが該当位置に移動します。

クロスリファレンスには次の特徴があります。

- クロスリファレンスの検索結果を削除できます。
- クロスリファレンスの検索結果をテキストファイルとして保存することができます。
- クロスリファレンスの検索結果を印刷できます。
- [動作環境] ダイアログの [動作設定] タブの下の [クロスリファレンスの結果の最大登録数] に検索結果の保持数の上限を指定できます。

ブックマーク

ナビゲータペインのブックマーク表示により、アプリケーション内でブックマークされたオブジェクトを見ることができます。

ブックマーク・オブジェクト

第 8 章 - GUI の拡張

ブックマークオブジェクトの定義により、ナビゲータに登録表示されるオブジェクトにリンクが作成されます。ユーザは、Ctrl + B の入力、あるいは [オプション] メニューの [ブックマーク] を選択することでブックマークを定義できます。

ブックマークの定義時にはデフォルトの名称が表示されますが、適当な名称に変更することができます。

ブックマークには次のような特徴があります。

- 作成順に表示されます。
- 新しく作成したブックマークオブジェクトが先頭行に表示されます。
- ブックマークから登録内容を削除できます。
- [動作環境] ダイアログの [動作設定] タブの下の [ブックマークの最大登録数] にブックマークの保持数の上限を指定できます。

MVCS

CTL 特性で [チーム開発を有効にします] のオプションを ON にすると、[ナビゲータ] ペインに、[MVCS] セクションが表示されます。ここでは、ログインユーザがチェックアウトしたオブジェクトの一覧が表示されます。

チェックアウトされたオブジェクトの一覧は以下のような特徴があります。

- 表示されているオブジェクト名をクリックすることで、そのオブジェクトにジャンプすることができます。
- [MVCS] セクション上でオブジェクトをチェックインすることができます。

第 8 章 GUI の拡張

[このページは意図的に空白になっています。]

Magic eDeveloper では、生産性向上のための機能強化が行われました。

この章では、次のトピックについて解説します。

- クロスリファレンス
- メインプログラム
- データコントロール
- モデル
- Magic フラットファイルでのアプリケーション実行
- タスク戻り値

クロスリファレンス

クロスリファレンスがサポートされるオブジェクト

次のオブジェクトに対してクロスリファレンスをサポートします。

- モデル
- テーブル
- カラム
- インデックス
- プログラム
- ヘルプ
- 権利
- メニュー
- タスクのリソース（入出力とフォーム）
- 式
- 変数

クロスリファレンスの表示

クロスリファレンスユーティリティを実行するには、調べたい登録オブジェクトにカーソルを移動して、Ctrl + X を押してください。続いて表示されるダイアログで、検索対象となるリポジトリを選択することができます。

クロスリファレンスの検索結果は、ナビゲータペインに表示されます。

メインプログラム

メインプログラムは、実行モードではアプリケーションが起動される時、また開発モードでは実行モードに切り替わる時に実行されます。メインプログラムには、アプリケーション全体の共通のリソースを定義できます。メインプログラムはデフォルトで登録されており、削除できません。

メインプログラムの実行モード動作

- メインプログラムは、実行モードでアプリケーションをオープンした時や、開発モードから実行モードに切り替わった時にバックグラウンドで自動的に動作します。また、メインプログラムは、プログラムリポジトリからプログラムを実行したりテーブルリポジトリから APG でプログラムを実行した場合にも動作します。
- メインプログラムには、メインテーブルがありません。バッチプログラムとして動作し、1 サイクル（タスク前処理、レコードメイン、タスク後処理）で終了します。
- メインプログラムに変数やフォーム、DB テーブル、イベント、ハンドラが定義されると、アプリケーション内のどこからでも参照することができます。

メインプログラムの開発モード動作

- メインプログラムは常にプログラムリポジトリ一覧の先頭にあります。
- メインプログラムはアプリケーションを新規に作ると自動的に登録されます。
- メインプログラムを開発者が登録することはできません。
- メインプログラムは削除できません。
- メインプログラムのタスクタイプはバッチに設定されていて変更できません。

メインプログラムとコンポーネント

メインプログラムは、コンポーネントとしては出力できません。

ホストアプリケーションのメインプログラムは、アプリケーションからのコンポーネントアクセスの前に、すでに読み込まれています。

コンポーネントが他のアプリケーションに読み込まれた場合、コンポーネントのメインプログラムは、アプリケーションのメインプログラムと他の全てのプログラムの間で実行されます。

メインプログラムでの関数

RUNMODE

今までのバージョンでは、開始 / 終了プログラムについては開発版 / 実行版の違いに基づいてプログラムが自動的に実行するかどうかを定義することができました。RUNMODE 関数を使用してエンジンの値を取得することで同じ

第9章 - 生産性の向上

機能を実現させることができます。

- マルチスレッドで最初にメインプログラムが実行された場合、「-1」が返ります。
- プログラムが実行エンジンで動作している場合は、「0」が返ります。
- プログラムが開発エンジンで動作していて、実行モードでアプリケーションをオープンしている場合は、「1」が返ります。
- プログラムが開発エンジンで動作していて、実行モードに切り替わった場合 (Ctrl + T) は、「2」が返ります。
- プログラムが開発エンジンで動作していて、F7 キーにより起動したり、APG でオープンした場合は、「3」が返ります。

PROG

現在のタスクのタスクパスを返します。全てのタスクはメインプログラムを含むため、返されるパスにはメインプログラムは含まれていません。

TDEPTH

実行時のタスクネストの深さを返します。メインプログラムはタスクのレベルとしては含まれていません。

データコントロール

今までのバージョンでは、コンボボックスやリストボックスのコントロールに対して、文字や式でしか設定ができませんでした。このため、カンマ(,)で区切られた複数文字列を含む、コントロール選択肢の文字列データを生成するサブタスクを作成しなければならない場合があります。

Magic eDeveloper では、コンボボックスやリストボックスをデータコントロールとして定義できます。これによって、データベースのフィールドから値の範囲を直接選ぶことができるようになりました。

コンボボックスとリストボックスの特性

コンボボックスとリストボックスのコントロールに、次の特性が追加されました。

- **ソーステーブル**.....テーブルリポジトリに表示されるテーブルの番号。
- **表示項目**.....選択されたソーステーブル内の項目番号。この項目の内容が表示されます。
- **リンク項目**.....選択されたソーステーブル内の項目番号。選択された表示項目に対応して、ここで指定したリンク項目の値が実行時に返されます。
- **インデックス**.....選択されたソーステーブルのインデックス名。テーブルインデックスから選択されます。
- **範囲**.....指定した項目の From/To の範囲を設定します。

モデル

モデルは、継承できる一組の特性で、特定のオブジェクトに定義できます。次のオブジェクトがモデルを定義できます。

- 項目
- コントロール
- フォーム
- ヘルプ

モデルで定義している特性が更新されると、そのモデルにリンクされたオブジェクト全てに、その特性が反映されます。ただし、オブジェクト側にて個別に定義しなおしている場合は除きます。

モデルリポジトリ

モデルリポジトリは、登録されているオブジェクトモデルを表示します。オブジェクトモデルは、各オブジェクトタイプのテンプレートとして利用することができます。各オブジェクトには、自動的にデフォルトの特性値が設定されます。オブジェクトモデルにリンクされたオブジェクトは、そのモデルに設定されている値と同じ特性値が設定されます。オブジェクトの特性値を個別に変更することで、変更した特性項目については、この継承関係がなくなります。

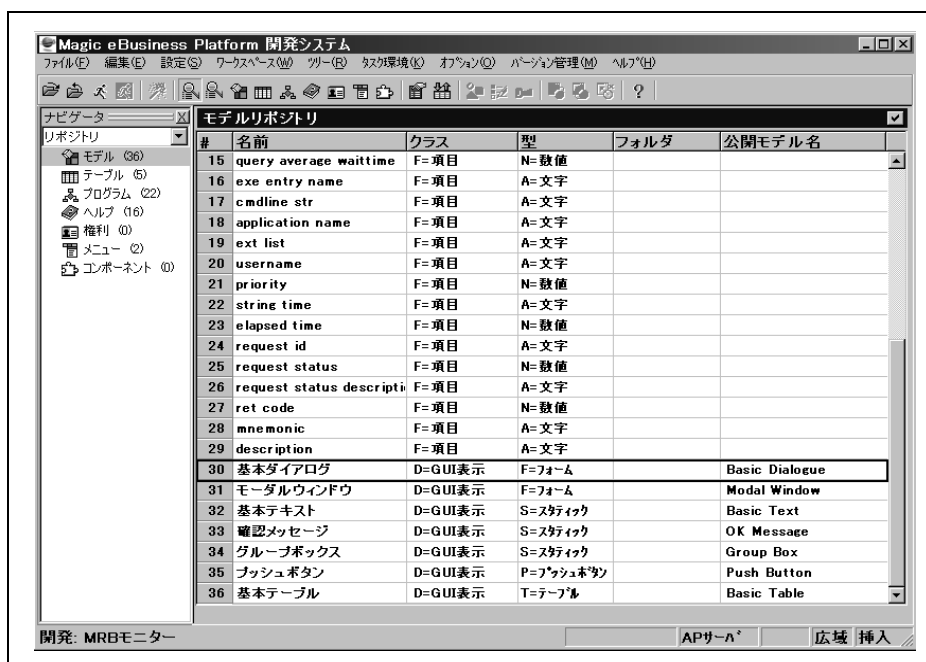


図 9-1 [モデルリポジトリ] テーブル

モデルリポジトリの属性

- 名前.....ユーザにより定義されるモデルの名前
- クラス.....モデルの対象となるオブジェクト、たとえば
 - 項目

第9章 - 生産性の向上

- ヘルプ
- フォーム/コントロール; ブラウザ形式、GUI表示、GUI出力、テキスト形式など
- 型.....各クラスが持っている、それぞれの下位属性
 - 項目
 - 文字.....文字列
 - 数値.....Int 型または、10進型の数値
 - 論理.....True または False
 - 日付.....数値型の日付の値
 - 時刻.....数値型の時刻の値
 - メモ.....可変長の文字列
 - BLOB.....Magic 外部で作成されたバイナリ情報
 - ヘルプ
 - 内部.....Magic アプリケーションの内部で定義されたヘルプ
 - Windows.....Magic アプリケーションの外部で定義されたヘルプ
 - フォーム/コントロール
 - 各インタフェースタイプのフォームとそのフォームで有効なコントロールが選択肢となります。

モデルを使用した作業

新しいオブジェクトは、デフォルトではシステムの既定モデルにリンクされます。アプリケーションが新しく作成されるとき、システムの既定モデルによって自動的に定義され、システムの既定モデルはデフォルトとして表示されます。



図 9-2 モデルの特性シート

特性

各クラスタイプに対して、特性のデフォルトが定義されます。デフォルトの値は黒色で表示されます。左側に表示される [継承解消] ボタンをクリックすることで特性の継承関係をなくすことができます。継承関係がなくなった特性は青色で表示されるようになります。なお、この色やフォントは変更できません。

モデル特性の継承解消

オブジェクトの特性値を個別に定義できます。特性値が個別に定義されたら、その特性項目については、リンクしているモデルの現在の値や更新された値を継承しなくなります。継承関係が解消された特性項目について、再度継承関係を復活させることもできます。

オブジェクトに対する異なったモデルの選択

モデルを他のものに置き換えた場合、リンクされたオブジェクトは新しいモデルの特性を継承します。個別に定義した特性は、システム既定のデフォルト特性への継承関係がなくなります。個別に定義した特性は、システム既定のデフォルト特性が更新されても影響されません。

Magic フラットファイルでのアプリケーション実行

Magic フラットファイル (MFF) によって、データベースから独立した形でアプリケーションファイルを実行することができます。バイナリファイルとして作成されたアプリケーションファイルは、ユーザコンピュータシステム上の任意の場所に配置できますが、実行版での利用に限定されます。

MFF はマルチプラットフォーム環境で使用することを想定しています。例えば、WindowsNT から UNIX へアプリケーションを移行する場合、リポジトリの入出力作業は必要ありません。

Magic フラットファイルの指定

MFF を指定する設定欄は、[アプリケーション特性] ダイアログで表示されます。開発者が「Yes」と設定すると、MAGIC.INI ファイルが更新され、MFF をロードするようになります。

アプリケーションの開発結果を MFF として出力することで、実行時には CTL ファイル用のデータベースは不要です。

Magic のアプリケーションは、MFF と圧縮を同時に指定できません。一方のパラメータを指定すると、他方が無効になります。

Magic アプリケーションを MFF として保存

既存のアプリケーションは、[ファイル] メニューの [MFF 形式で保存] をクリックすることで保存することができます。MFF で保存された Magic アプリケーションは、開発版ではオープンできません。MFF 形式ファイルは実行

第9章 - 生産性の向上

版でのみ利用できるように設計されています。

タスク戻り値

Magic eDeveloper の [タスク特性] には、[戻り値] という新しい特性が追加されました。これは、値か式で指定されます。プログラムの処理が終了すると、この値が返ります。

CALLPROG () という新しい関数が追加され、プログラムはこの関数のパラメータに式などにより記述することで起動されるようになります。この関数の戻り値としては、起動されたプログラムの戻り値の値が返されます。

第9章 生産性の向上

[このページは意図的に空白になっています。]

Magic eDeveloper には、開発時に便利な多くの新機能が追加されています。

この章では、次のトピックについて解説します。

- パラメータ整合
- If-Else ブロック
- コメント
- 参照整合性
- 条件リンク

パラメータ整合

今までのバージョンでは、変数とパラメータを区別することができませんでした。Magic eDeveloper では、特定の変数をパラメータとして宣言できるようになります。この変数は、コールコマンドにおいて使用すべき引数として認識されるようになります。

セレクトコマンドとパラメータ

セレクトコマンドの選択肢の1つとしてパラメータが追加されました。これにより、セレクトコマンドは次の3つのオプションから選択できるようになります。

- 実データ
- 変数
- パラメータ



パラメータは、他のタスクから渡される引数を受け取ります。タスクにパラメータが定義されていない場合、変数に引数の内容が渡されません。

パラメータテーブル

パラメータは、[タスク環境 / 変数項目] の [パラメータ] テーブルで定義します。

パラメーター一覧

コールコマンドで使用するパラメータが、[パラメーター一覧] として表示されます。

パラメータのタイプは、コールプログラムコマンドで表示される [パラメーター一覧] に反映されています。[パラメーター一覧] には、呼ばれるプログラムに定義されているパラメータの数と詳細が表示されます。

パラメータを定義した場合のプログラム呼出

コールコマンドによってパラメータが定義されているプログラムやタスクを呼び出す場合、ズームによって [パラメーター一覧] を表示し、呼ばれるプログラムやタスクのパラメータ内容を表示させることができます。この内容に合わせて引数の追加や削除を行い、定義されたパラメータに合わせるすることができます。

パラメータを定義しない場合のプログラム呼出

コールコマンドによって、パラメータが定義されていないプログラムやタスクを呼び出す場合、[パラメーター一覧] には、何も表示されません。パラメータとして指定する項目は、最大 255 までの制限があります。パラメータは、呼ばれるタスクの変数に登録されている順に引き当てられます。

If-Else ブロック

ブロックコマンドに Else オプションが追加されました。今までは、個々の条件毎に新しくブロックを作成しなければなりませんでした。Magic eDeveloper では、If-Else ブロックを指定することで、最初の条件に対する False 条件の処理を含め、1 つのブロックコマンドで定義できるようになりました。

また、If-Else ブロックはネストさせることができます。

コメント

Magic eDeveloper では、アプリケーション内のオブジェクトにコメント文を添付させることができます。

第10章 - 開発機能の拡張

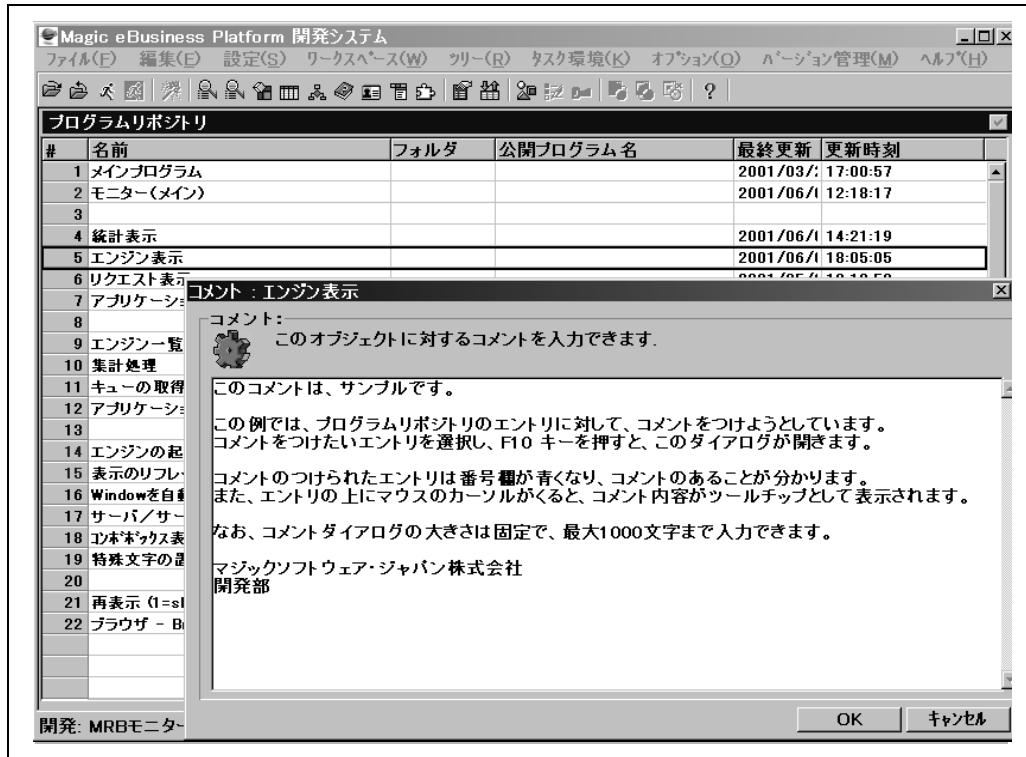


図 10-1 コメントダイアログ

コメント文を添付できるオブジェクトは次の通りです：

- モデル
- テーブル
 - テーブル
 - カラム
 - インデックス
 - 外部キー
- プログラム
 - タスク
 - 変数項目（変数、パラメータ）
 - 入出力テーブル
 - フォームとコントロール

第 10 章 開発機能の拡張

- ハンドラ
- ヘルプ
- 使用権利
- ユーザメニュー
- コンポーネント

参照整合性

SQL では、外部キーを使用してテーブル間のリンクを定義することができます。外部キーはテーブル内のカラムをもとに定義し、他の SQL テーブル内で定義された参照キー（プライマリキーまたはユニークキー）に対応するようにします。

テーブルのカラムに対して外部キーを設定することで、それに対応している参照キーのカラムとの参照整合性を保持できます。例えば、メインテーブルとして定義されているテーブルの外部キーまたはテーブルを削除することなく参照キーを削除することはできません。

Magic eDeveloper では、テーブルリポジトリでプライマリキーと外部キーを定義できます。

テーブルリポジトリ

プライマリキー

[プライマリキー] のチェックボックスが、[インデックス特性] に追加されました。プライマリキーは 1 つのインデックスのみを指定できます。このインデックスは、ユニークかつ Null 値許可「No」でなければなりません。

外部キー

[外部キー] のカラムが、テーブルリポジトリに追加されました。[外部キー] でズームすると、[外部キー] テーブルが表示されます。この [外部キー] テーブルには、2 つのテーブルがあります。外部キー定義テーブルと外部キーセグメントテーブルです。

外部キー定義テーブルには次の内容が表示されます。

- **外部キー番号**.....自動的に割り振られた、外部キーの順番を意味する内部番号。
- **外部キー名**.....Magic で使用する外部キーの名前。データベースに外部キーが作成される場合、この名前になります。
- **参照テーブル**.....データベースに作成された任意の Magic のテーブルから選択します。テーブルリポジトリに登録されたテーブルのみ参照できます。
- **参照キー**.....参照されたテーブルから特定のインデックスを選択します。
- **DB に作成**.....チェックボックスにチェックした場合は、外部キーがデータベースに作成されます。

第 10 章 - 開発機能の拡張

外部キーセグメントテーブルには次の内容が表示されます。

- **現在のテーブルのセグメント番号**.....自動的に割り振られた、現在のテーブルのセグメントの順番を示す内部番号。
- **現在のテーブルのセグメント**.....カラムでズームすることにより現在のテーブルからこれらのセグメントを選択できます。現在のテーブルの各セグメントは、参照するテーブルに合ったセグメントを持っていない限りありません。
- **参照テーブルのセグメント番号**.....自動的に割り振られた、参照するテーブルのセグメントの順番を示す内部番号。
- **参照テーブルのセグメント**.....参照されるテーブルで定義されているインデックスを選択した場合、これらのセグメントは自動的に表示されます。参照キーを別のものに変更すると、表示されているセグメントの内容が書き換わります。

APG

外部キーを持つテーブルで APG を実行するとき、生成されるプログラムにはそのテーブル内のフィールドが定義されます。これは今までのバージョンと同様です。

Magic eDeveloper では、[APG] ダイアログに、テーブルに定義された全ての外部キーの一覧を表示するリンクタブが追加されました。それぞれの外部キーはその順に番号がつけられていますが、この番号を修正することでキーの順序を変更することができます。番号に 0 を設定すると、その外部キーは対象外となります。

外部キーを選択することにより、[選択カラム一覧] に、参照テーブルのすべてのカラムが追加されます。[選択カラム一覧] では、フィールドを削除できますが、参照テーブルのプライマリキーセグメントや、参照しているテーブルのプライマリキーセグメントを削除することはできません。

外部キーを選択し、その順序を決定することにより、プログラムにそれぞれの外部キーに対するリンクコマンドが追加されます。

参照テーブルに対するリンクは、レコードメインの処理テーブルの最後に表示されます。またそれぞれのリンクには次のコメントが表示されます。

外部キーによるリンク : < 外部キーの名前 >

もしメインテーブルと参照テーブルが同じ SQL データベースであれば、結合リンクが定義されます。それ以外の場合は、参照リンクになります。

APG では、メインテーブルの外部キーのみを検索します。参照テーブルに定義された外部キーをネストすることはできません。

リンク条件

今までのバージョンでは、リンクコマンドの条件では、実際にレコードを読み込むか否かの指定を行うことはできませんでした。Magic eDeveloper では、リンクの実行条件を指定できるようになりました。

開発モード

タスクのフローテーブルにおいて、[条件] 欄に「Yes」か「No」または「式」

第 10 章 開発機能の拡張

を定義することで、リンク条件を設定することができます。

[条件] 欄に「Yes」を設定した場合、または「式」の結果が「True」となる場合、リンクコマンドによって該当するレコードを読み込みます。

[条件] 欄に「No」を設定した場合、または式の結果が「False」となる場合、該当するレコードは読み込みません。レコードはリンクが失敗した場合と同じようになり、次のような結果になります。

- リンクコマンドによってセレクトされたカラムの全ての値は、デフォルト値になります。
- リンクの戻り値は、「False」となります。
- 条件が「True」と評価されるまで項目更新は行われません。
- リンク条件がレコード後処理の終了するタイミングで「False」と評価された場合、レコード前処理以降に読み込まれたリンクレコードに対する修正は無効になります。

ユーザインタフェース

[リンク特性] ダイアログに新しい設定項目が追加され、リンク条件の評価時期が指定できます。

リンク特性ダイアログ

条件パラメータが新しいリンク条件の機能になったため、今までの確認モードのリンク指定は、[リンク特性] ダイアログの[確認] 欄で設定するようになりました。

この「確認」のリンク指定は、照会リンクと結合リンクで有効です。

照会リンクと「確認」のリンク指定の組合せは、削除された確認モードのリンク指定に取って代わるものになります。

以前のバージョンから移行すると、確認モードのリンクは、「確認」のリンク指定が有効な照会リンクに変換されます。

リンク条件の評価 特性

[リンク条件の評価] の設定が「T=タスク」になっている場合、データビュー全体を読み込む前に 1 回だけリンク条件の評価が行われます。リンク条件が「False」と評価された場合、リンク部のセレクトコマンド行は評価されません。結合リンクの場合、SQL 文は結合部の記述が含まれません。

結合リンクや外部結合リンクに対してこの設定を「r-レコード」とし、リンク条件が「False」と評価された場合、リンクデータはあるなしに関わらず読み込まれ、リンクフィールドの値はデフォルト値に置き換えられます。リンクの再計算においては、結合リンクと外部結合リンクは通常の照会リンクと同じように動作します。