Starting LANSA

「Starting LANSA」研修テキスト - Version V14

ご注意

- 本文書内の情報が正確かつ完璧であるよう、細心の注意を払っておりますが 、誤植または技術上の誤りがある場合もございます。
- (株)ランサ・ジャパンは、本文書使用の結果生じたお客様の損失に対しいかなる責任をも負うものではありません。

弊社Webサイトへもぜひアクセスしてください

URLは http://www.lansa. jp

株式会社 ランサ･ジャパン

当コースの目的と内容

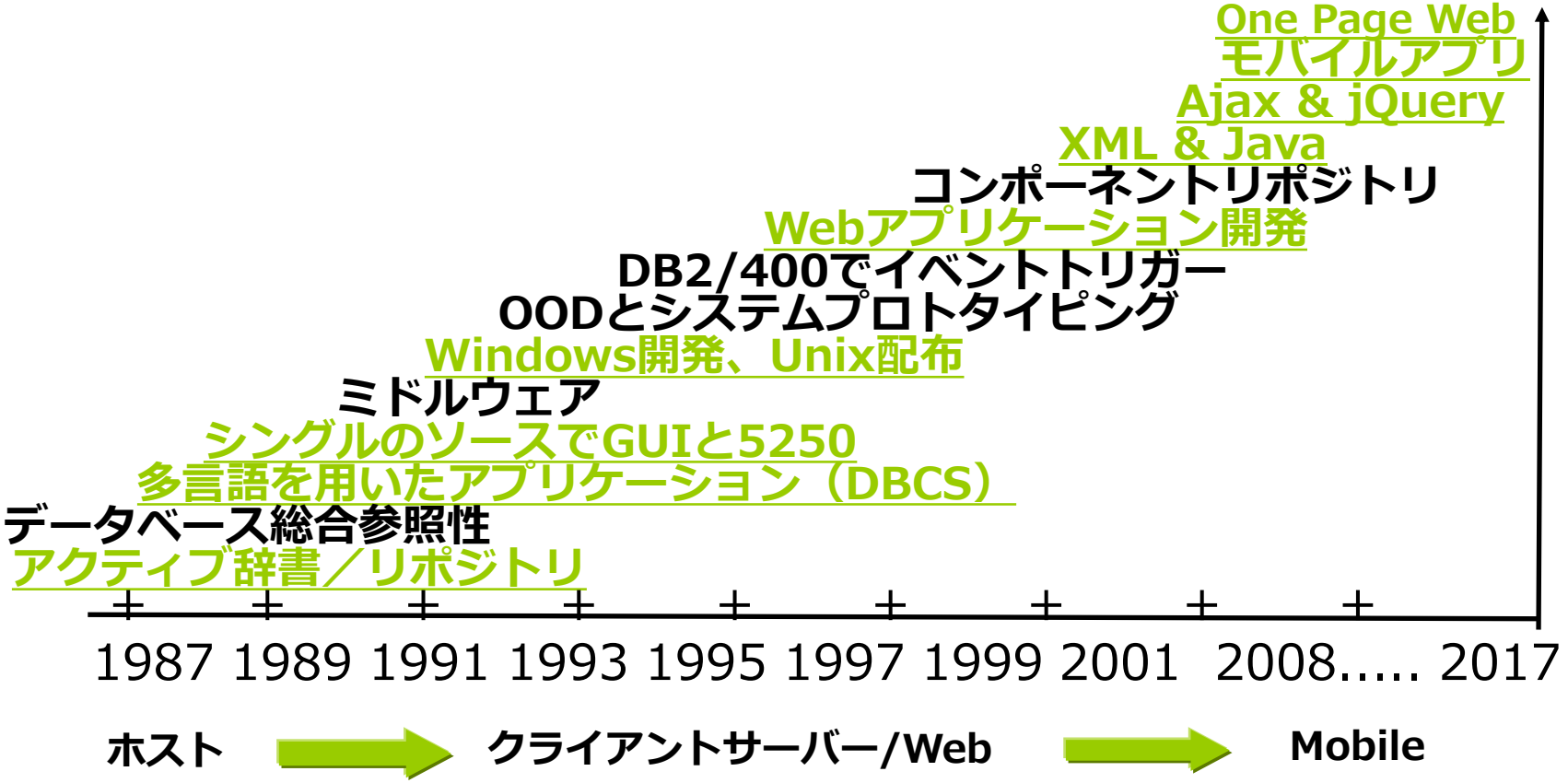
LANSA

LANSAで開発を始めるにあたり必要となる前提知識を学ぶことを目的としています。

- LANSA とは
- アプリケーションのタイプ
- IDE
- リポジトリ基礎
  a. フィールド定義
  b. ファイル定義
  c. 論理ビュー定義
    - 演習：フィールド/ファイル/論理ビューの作成
- RDMLとテンプレート
    - 演習：プロセス/ファンクションの作成
- 開発環境の構築と運用
    - 演習：チェックイン/チェックアウト
- 区画
- RDMLとRDMLX

LANSAとは…

LANSAの製品進化の歴史



CASE から、開発実行環境へ

LANSA

CASE

誰でも
一定の品質の
できるだけ早く

プログラム作成を

支援する

開発・実行環境

システムを

様々なテクノロジーを
一つの思想体系の元に

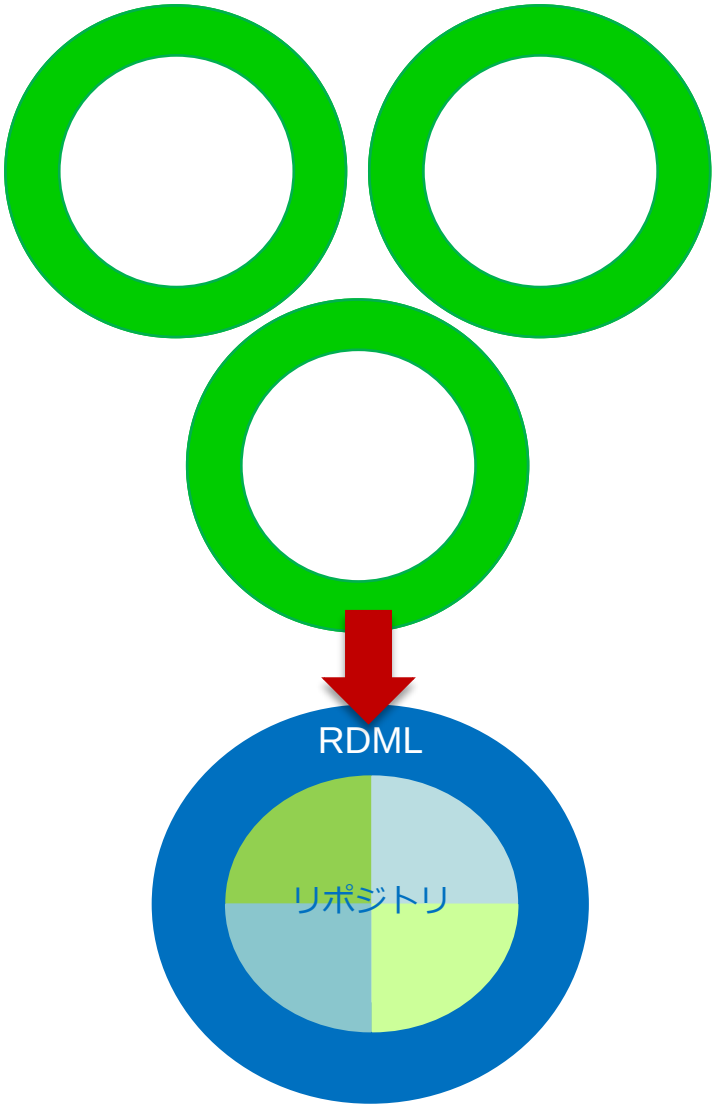
開発/実行する

LANSA の思想

Advanced Software Made Simple

- リポジトリ・テクノロジー
  - システム情報の一元管理
- 第４世代言語
  - 読みやすく、プラットフォームに依存しない言語　RDML
    （Rapid Development and Maintenance Language）
- オブジェクト・アクセス・モジュール(OAM)
  - リポジトリから生成されるプラットフォームに依存しないデータベースアクセスモジュール

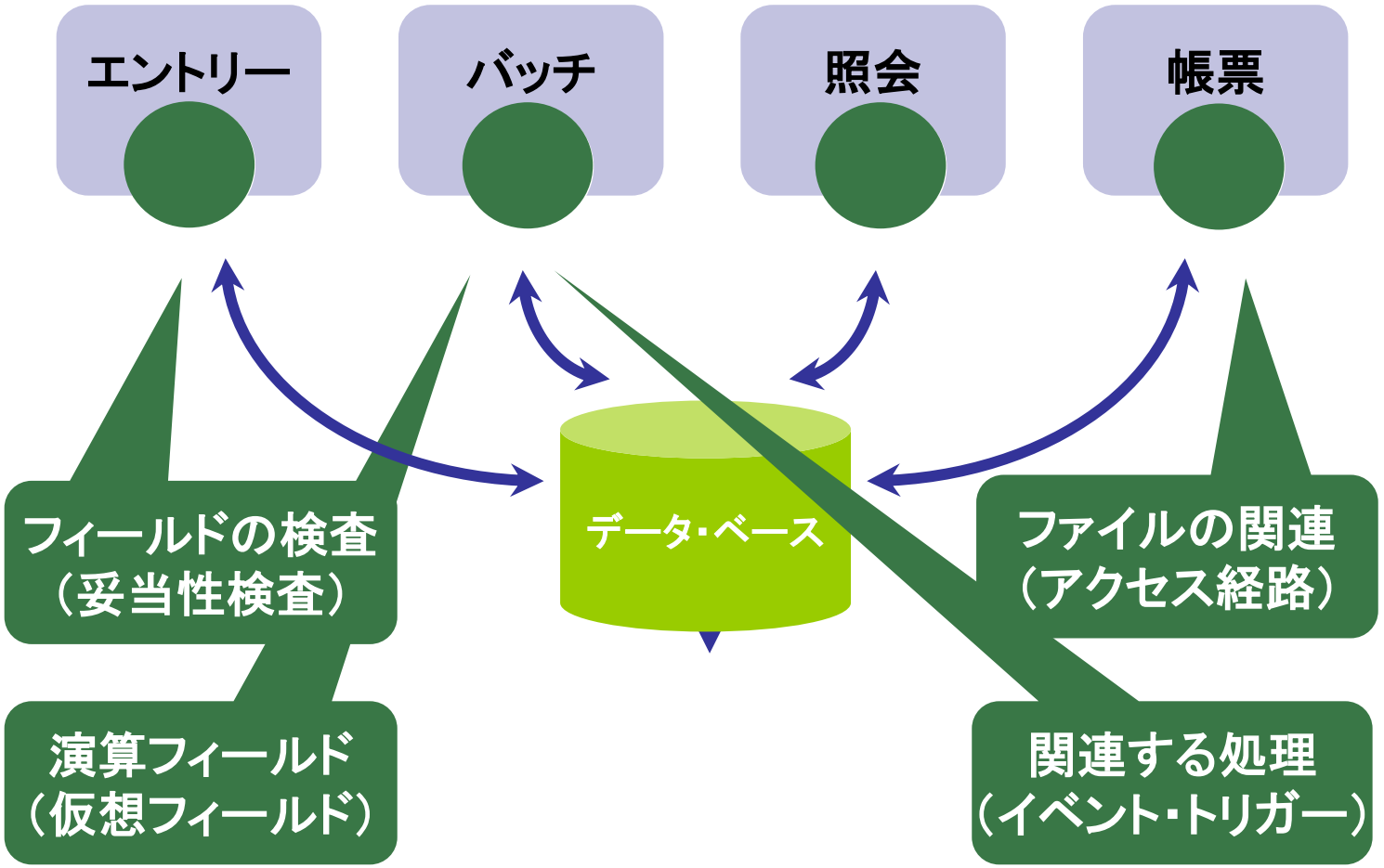
情報の一元管理



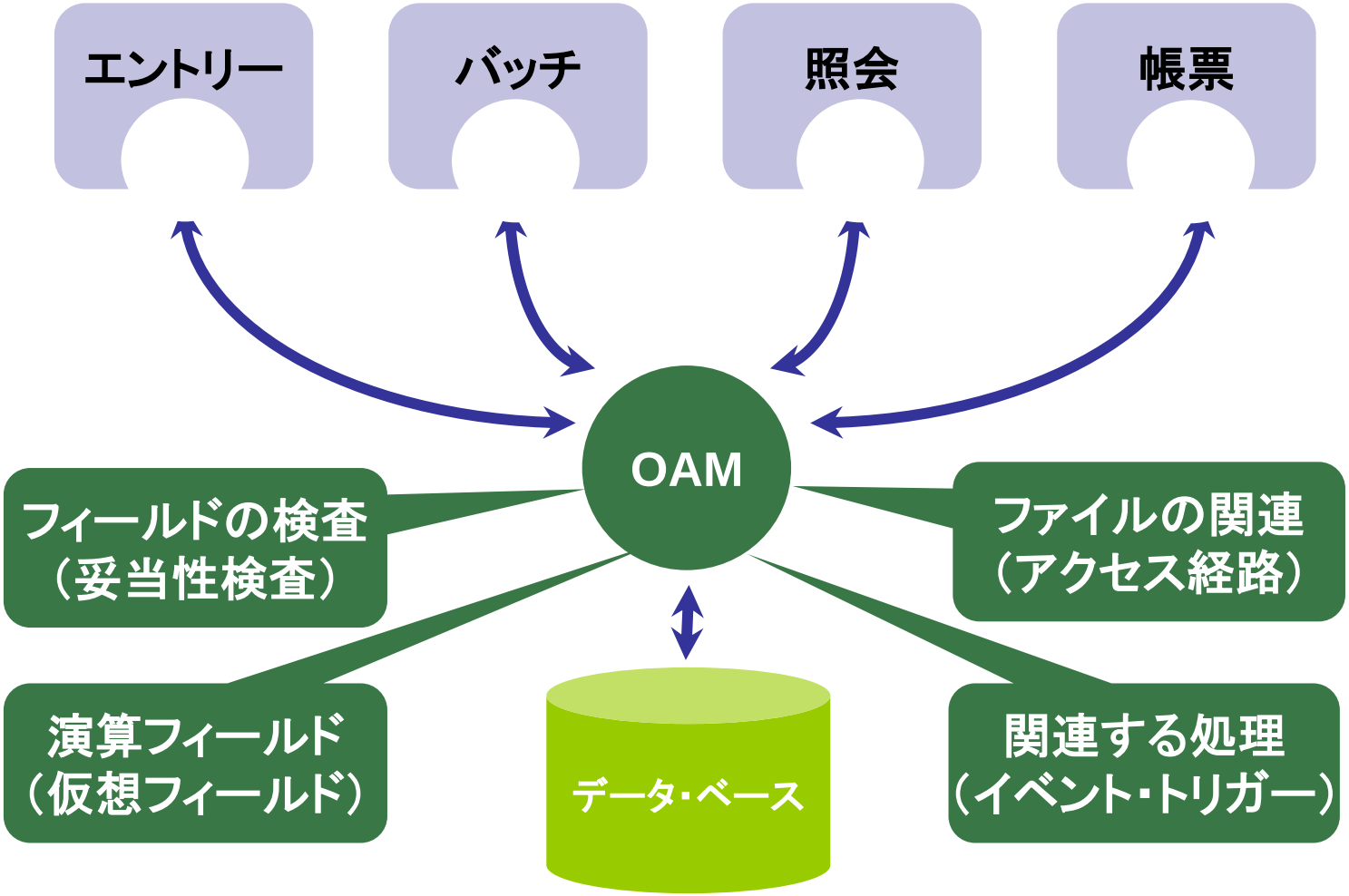
多くの場合、プログラムには、
アプリケーションに関する
重複した情報が含まれています。

LANSA リポジトリは、これらの
重複した情報をプログラムから
消去し、1ヶ所で一元管理します。

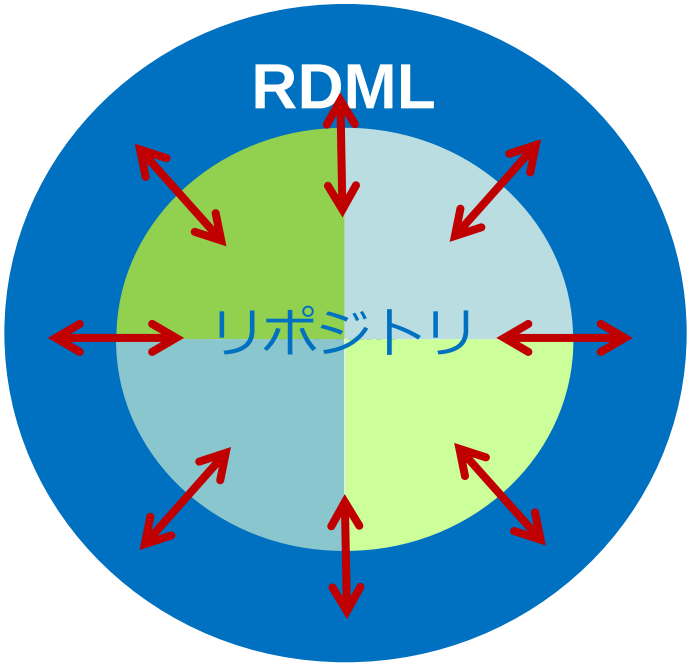
従来のビジネス・ロジック管理



LANSAのビジネス・ロジック管理



一緒に動く

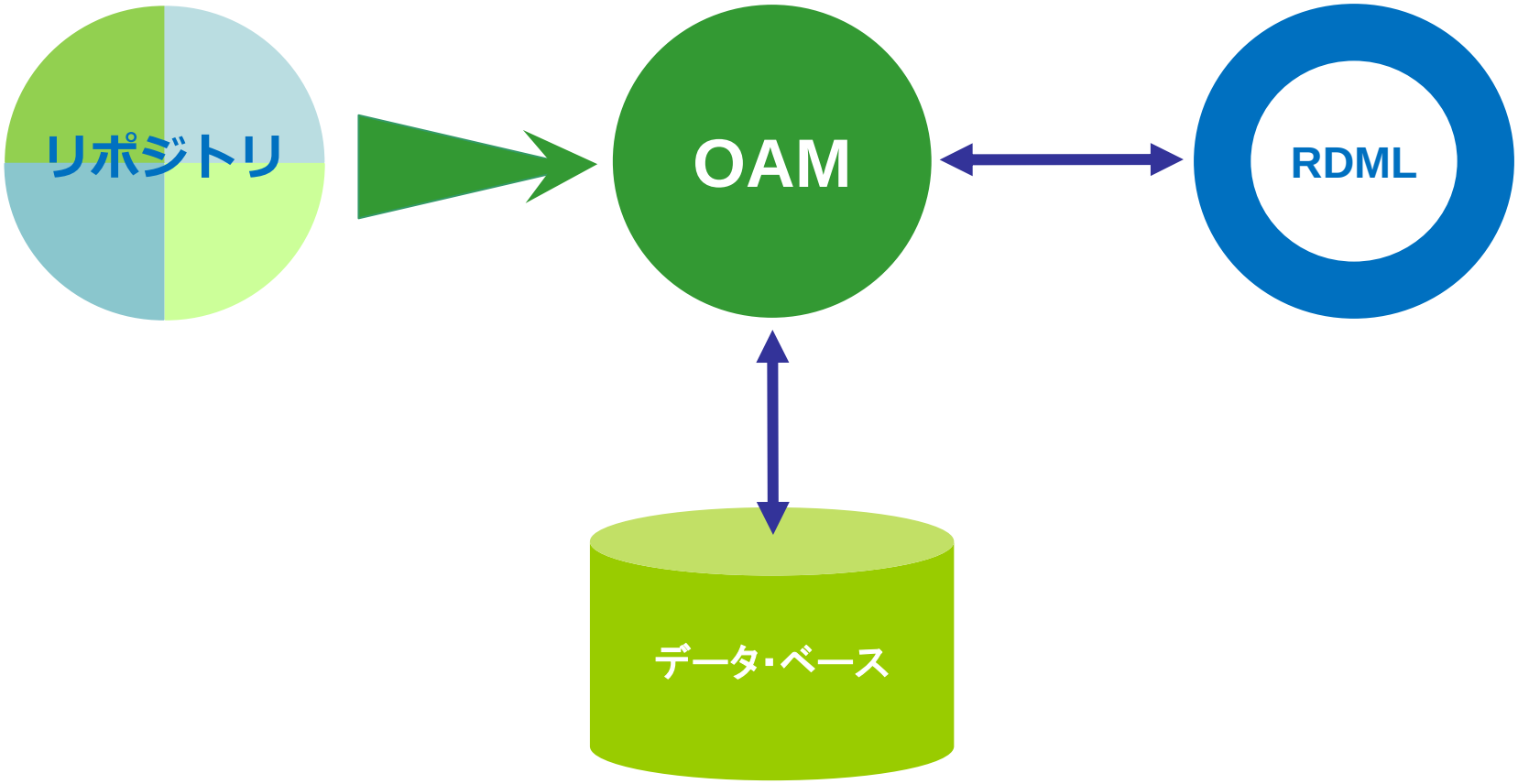


ステップ1.リポジトリの作成

ステップ2.RDMLの作成

・ LANSA リポジトリと RDML は、一緒に働きます。
・ アプリケーションの作成と実行には、両方のコンポーネントを使用します。

LANSA の構造

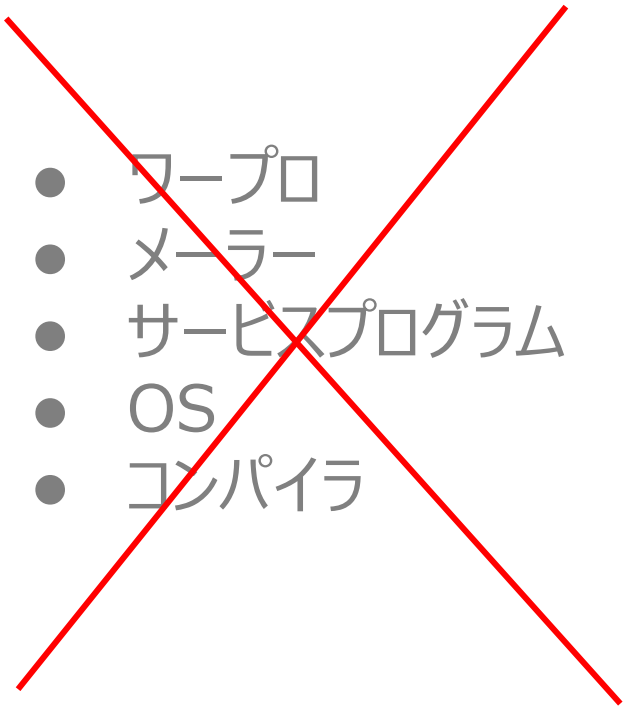


LANSA で作成するシステム

ビジネス・アプリケーションを作成する事に

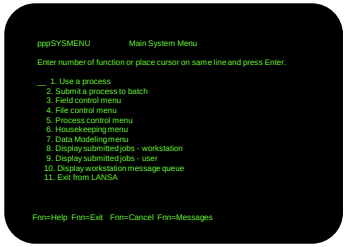
最適化されています。

- **会計システム**
- **販売管理システム**
- **生産管理システム**
- **通販のWEBサイト**
- **予約システム**

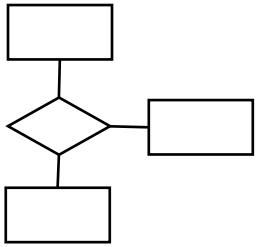


今までのプログラムは・・・

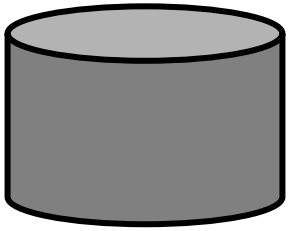
インターフェース



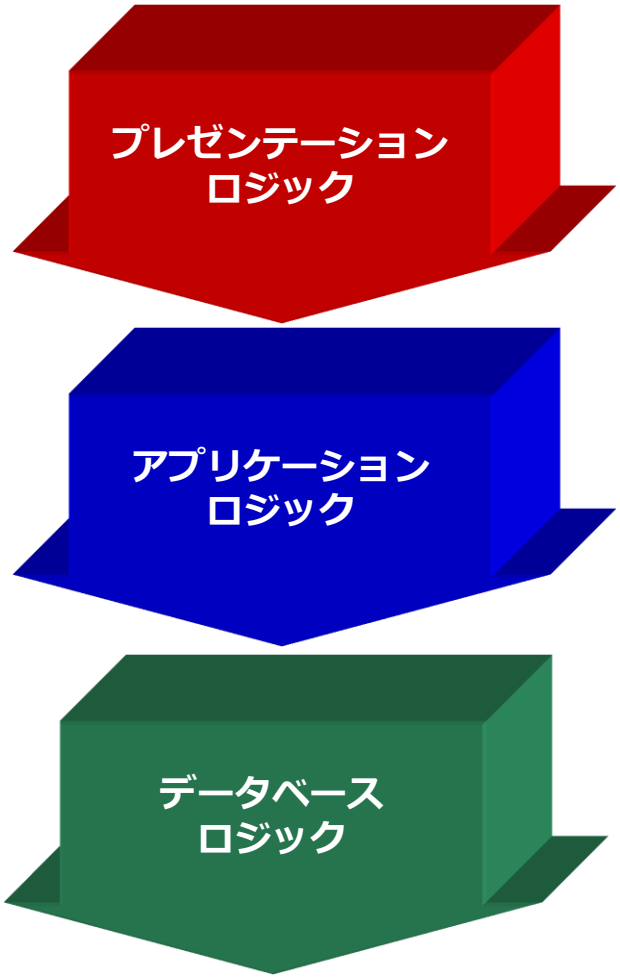
プログラム



データベース



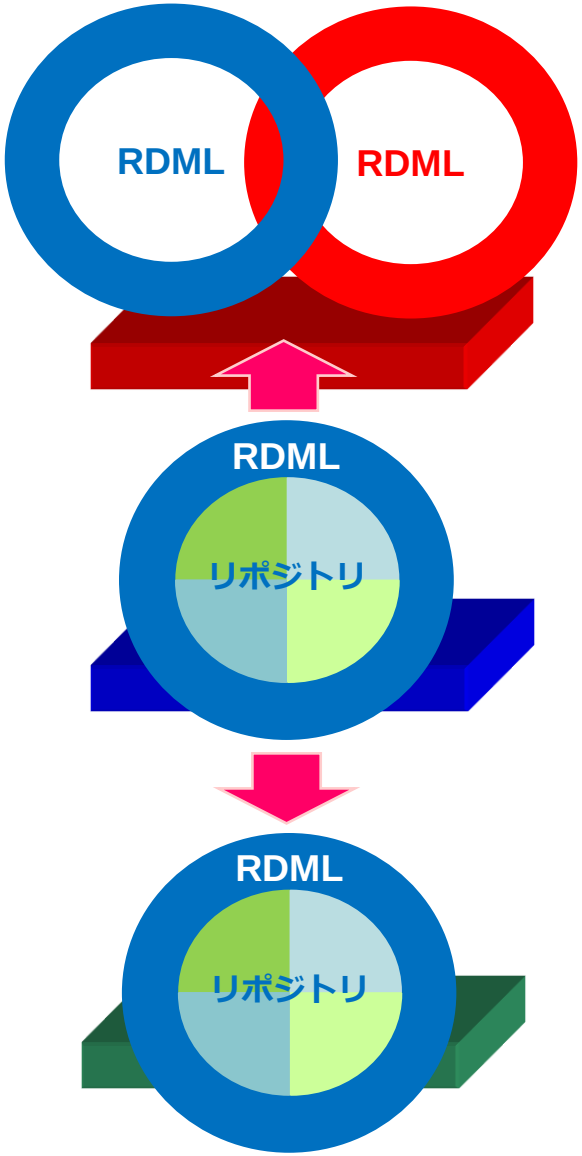
LANSA は・・



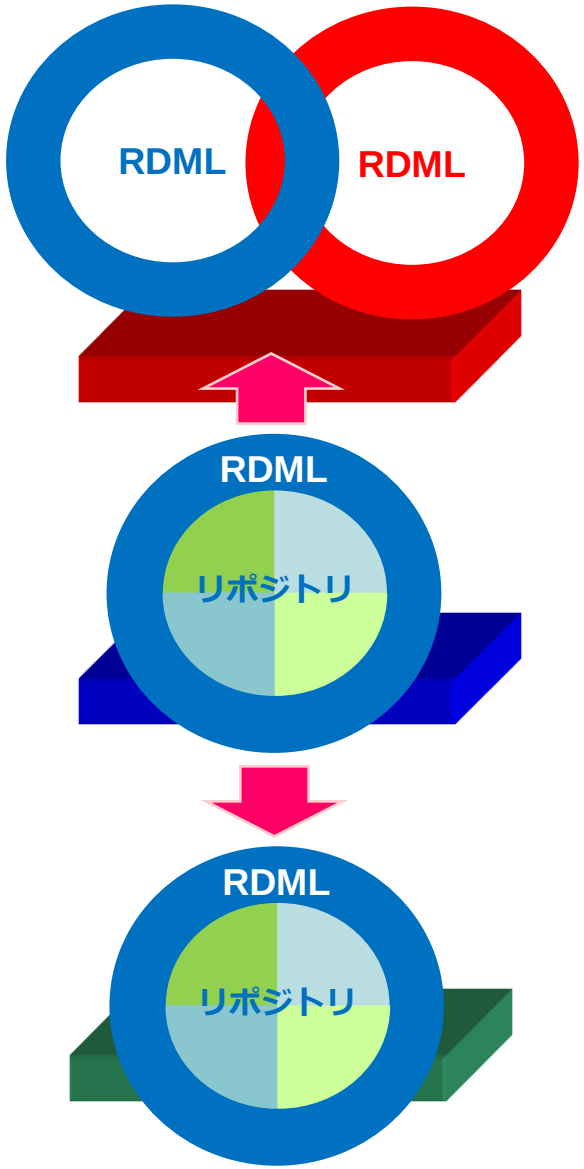
Windows?
Mac?
OS/400?
AIX?
etc.

C/C++?
Java?
HTML?
XML?
RPG?
Cobol?
etc.?

DB2/400?
Oracle?
SQL Server?
etc.



LANSAのソリューション

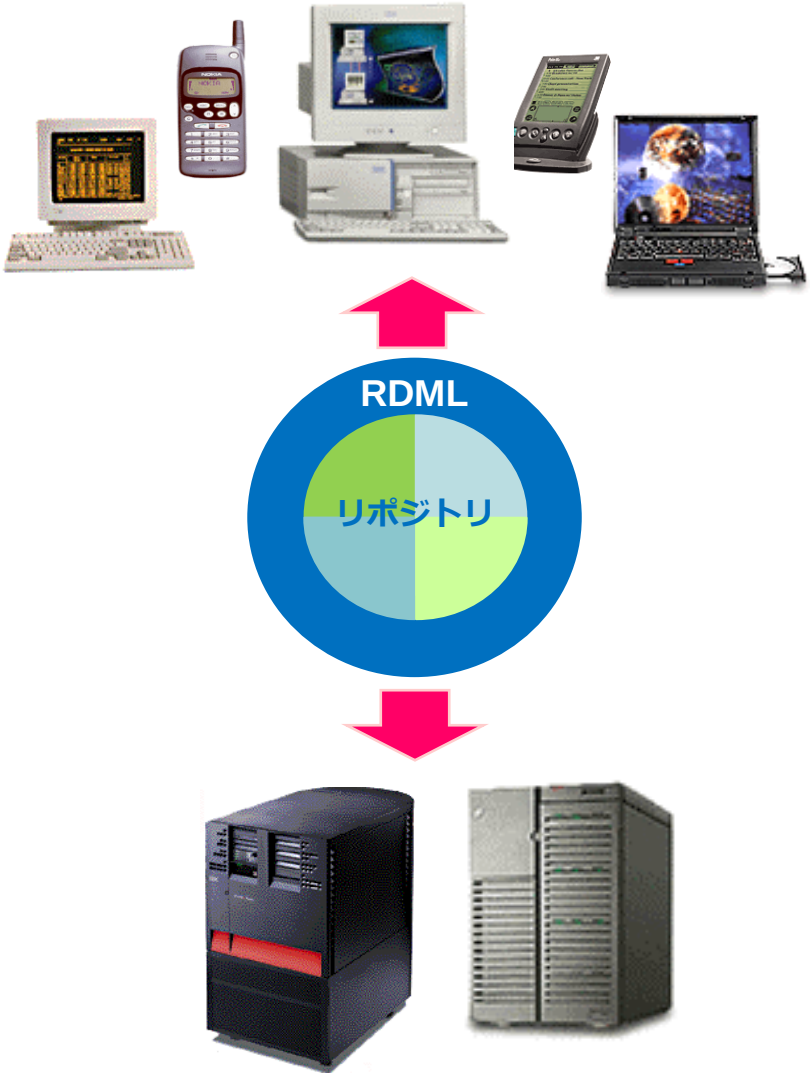


プラットフォームに依存しない
アプリケーション・アーキテクチャ

一元管理される
アプリケーション定義

ポータブルな
データベース・アーキテクチャ

LANSAはアーキテクチャーです



Web アプリケーション
Windows アプリケーション

1 つの
コード・セット

クライアント／サーバー
アプリケーション

1 つの
スキル・セット

ホスト・アプリケーション
（AS/400、AIX、HPUX等）

アプリケーションのタイプ

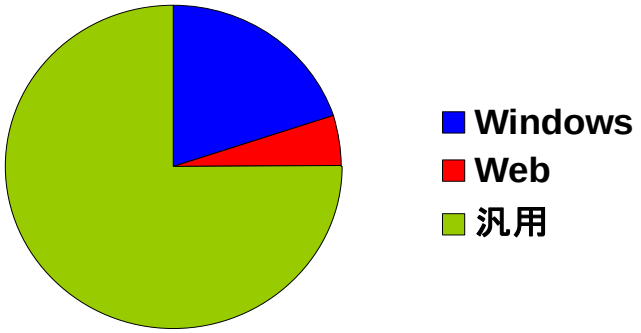
開発のタイプ

以下のような開発のタイプが考えられます：

- パッケージ／既存アプリケーションの拡張
- ビジネス･アプリケーションの再構築
- 新規アプリケーションの開発

アプリケーションのタイプも考慮する必要があります：

- 汎用
- Windows 最適化
- Web 最適化



アプリケーションのタイプ

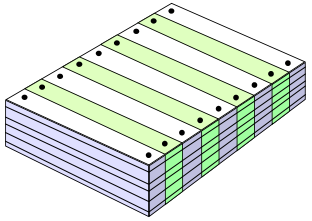
LANSA

プロセスドリブン

大量のデータエントリー

迅速な単一作業

汎用インターフェース



カスタマードリブン

多種多様のオフィス任務

電話の割り込み

手紙を書く／グラフを分析する

頻繁なタスク切り換え

オフィス・ツールのインテグレート

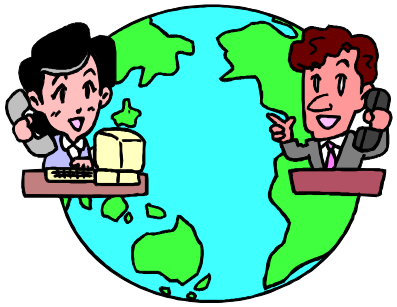
Windows 最適化インターフェース

顧客からのダイレクトインターネットアクセス

時たまの使用

頻繁なタスク切り換え

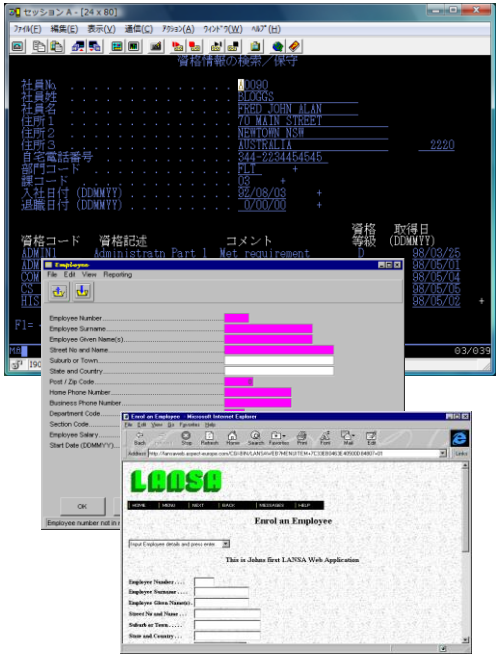
Web 最適化インターフェース



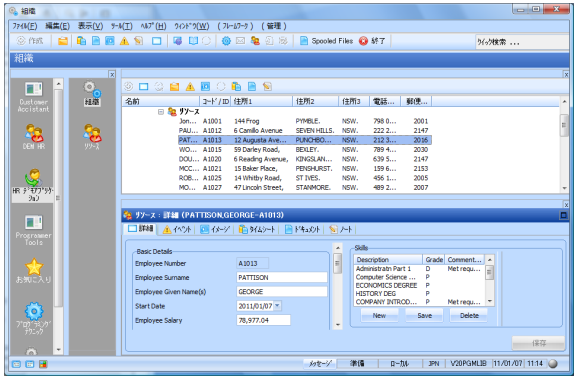
アプリケーションのタイプ

LANSA

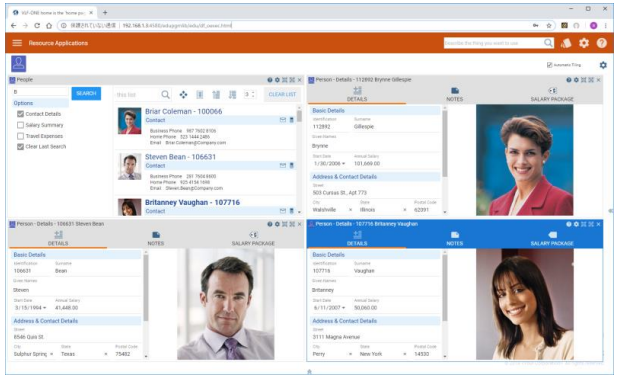
汎用インターフェース



WINDOWS最適化



WEB最適化

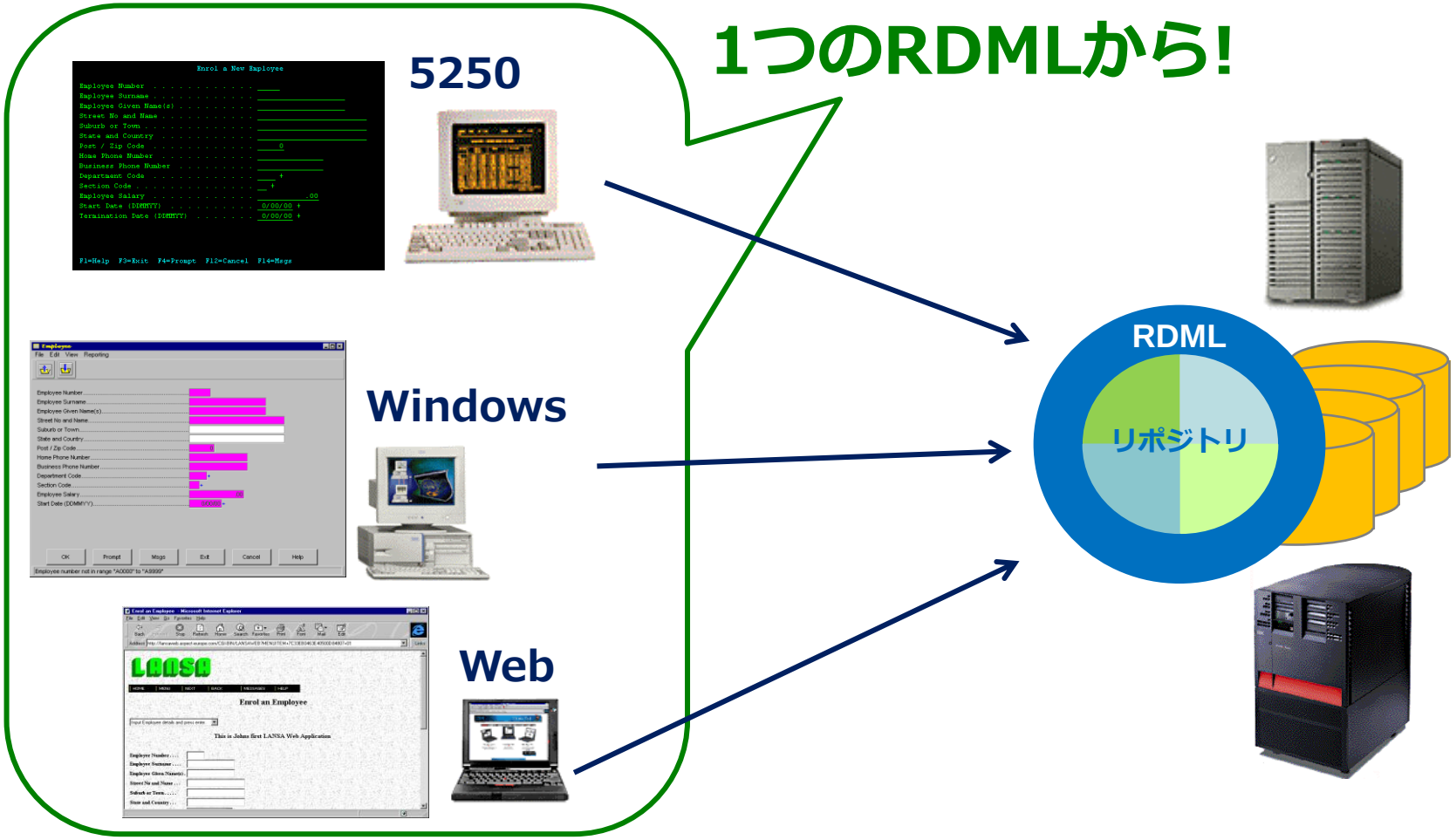


Function

Visual LANSA Form

Visual LANSA Web

Function(ファンクション)



様々なプラットフォームで稼動する事を前提とした、
汎用インタフェースです。

ファンクションのメリットと注意事項

LANSA

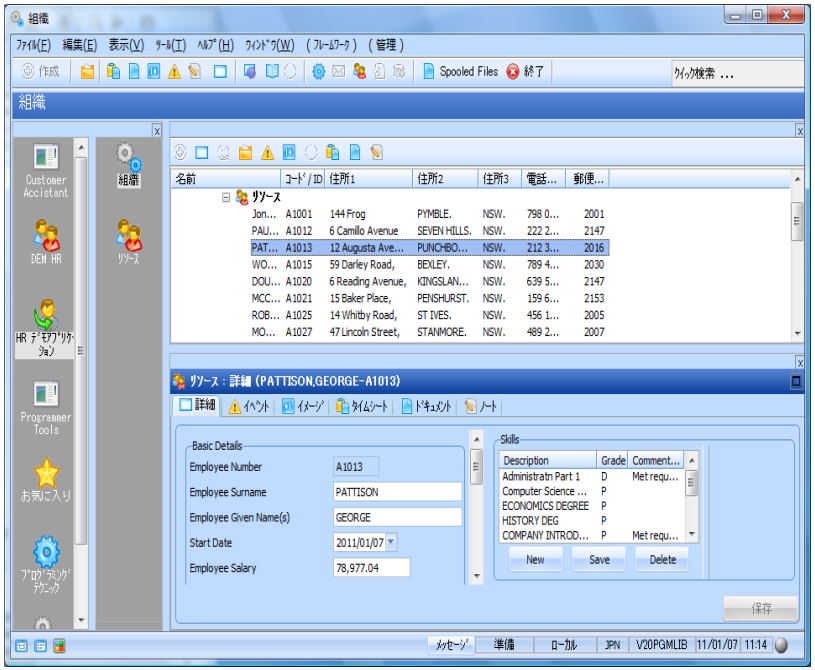
メリット

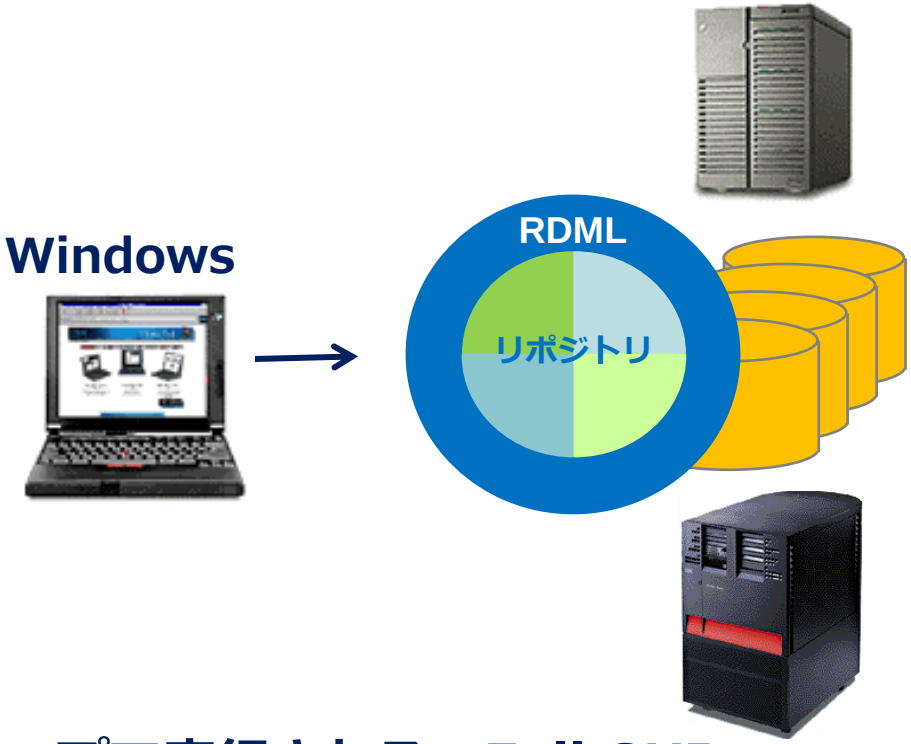
- ソースコードを変更することなく、様々なプラットフォームで実行することが可能です。
- 「伝票を入力する」という目的では必要十分な機能を提供しています。

注意事項

- 5250での実行を前提しているため、GUIであっても80x24の画面サイズを超えることができません。
- Webの実行は、永続セッションで実行されるため、HTTPの規約に反しています。※悪性要素ではない

Visual LANSA Form





Windowsデスクトップで実行される、Full GUI のアプリケーションです。

Formのメリットと注意事項

LANSA

メリット

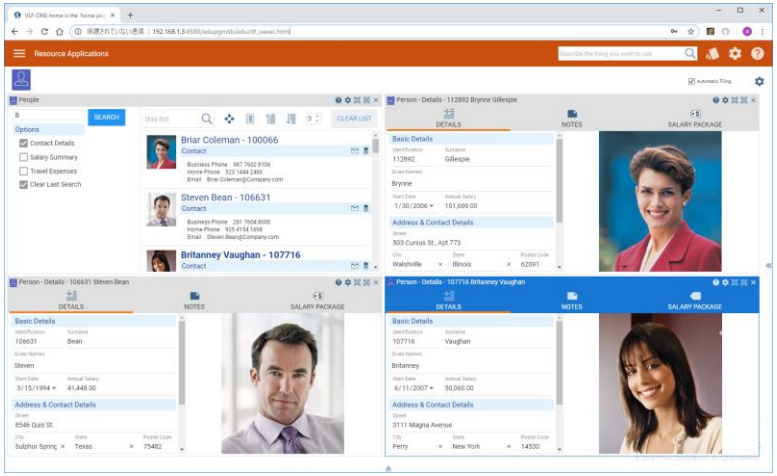
- 画面の制約がありません
- ユーザーに親和性のあるインフェースを提供できます。
- ActiveXを使用することができます
- Officeとスムーズに連携ができます

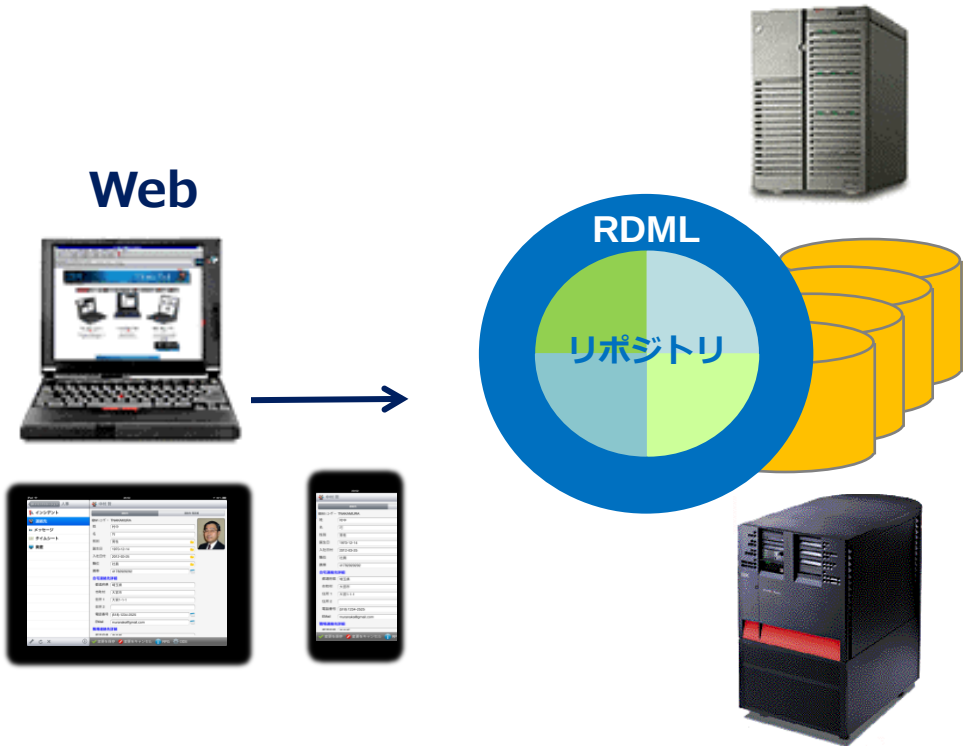
注意事項

自由であるが故に…

- 設計者、プログラマーのスキルの差が顕著になります。
- GUIにもかかわらず、操作性が悪くなることがあります。
- GUIで必要な、操作の一貫性を失う場合があります。
- アプリケーション・モジュールの配布が必要となります。

Visual LANSA Web





マルチ・デバイスに対応した、JavaScriptとJSONで表現されるWebアプリケーションです。

VL Webのメリットと注意事項

メリット

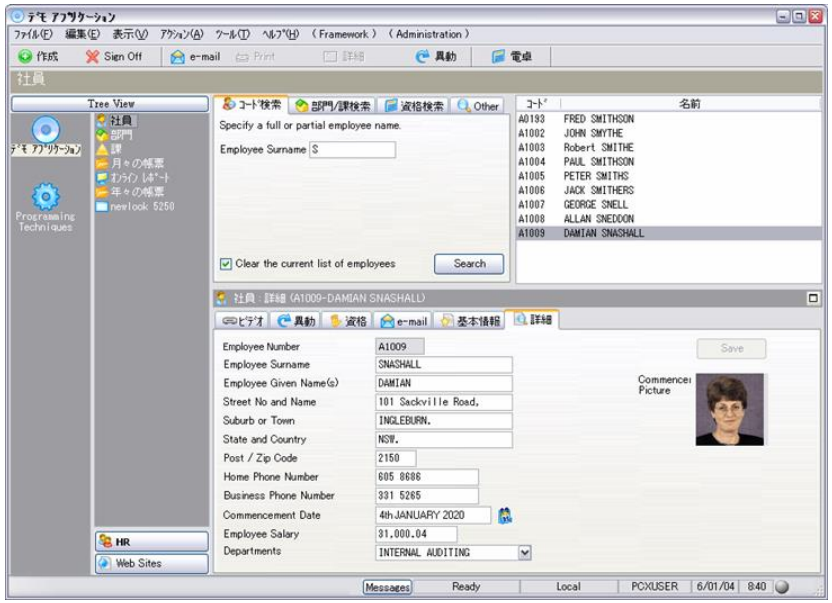
- RDMLXで、WebデザインとDB I/Oの両方のロジックを作成できます。
- Formと同等のデザイン手法とコンポーネント・テクノロジーで開発ができます。
- C/Sと同等のパフォーマンスを実現します。
- マルチデバイスに対応した、マテリアル・デザイン・コントロールが用意されています。

注意事項

Webであるが故に…

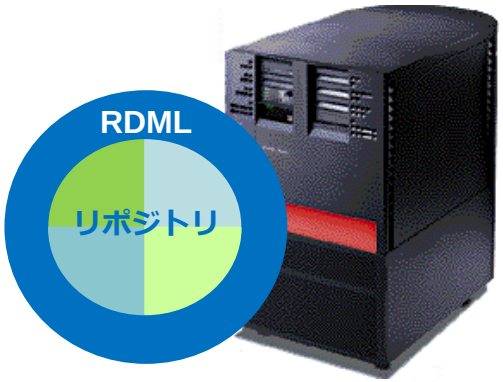
- セッションの永続性がありません。(レコードロックがかかりません。)
- URLを解析された場合に、そのプログラムを直接呼び出すことができてしまいます。

RAMP(Rapid Application Modernize Process)



Windows

or



Web

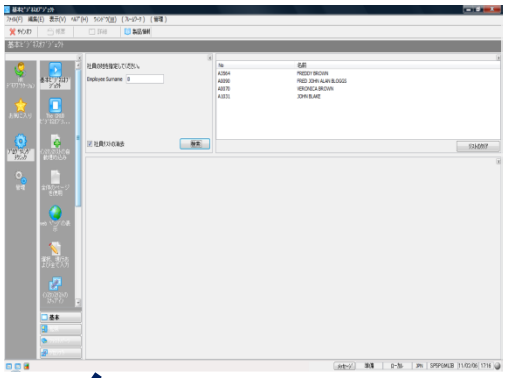
WindowsまたはWebブラウザーで実行される、フレームワークのアプリケーションです。

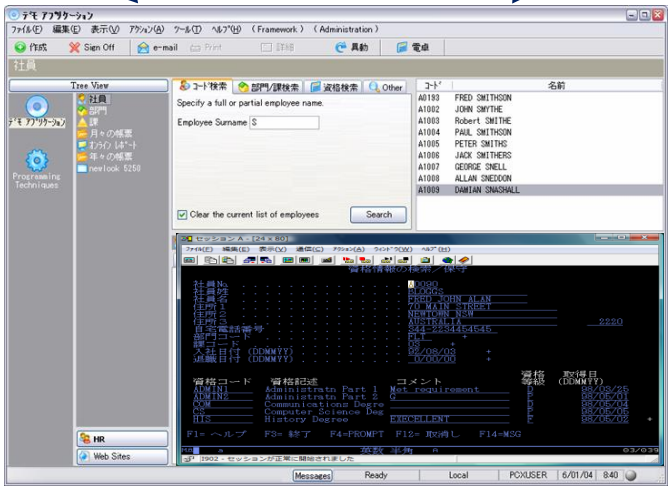
RAMPの仕組み

LANSA

5250

GUI





Visual LANSAのフレームワークに、5250をはめ込んで実行します。

RAMPのメリットと注意事項

メリット

- 5250アプリケーションをGUIに変換できます。
- 5250アプリケーションを修正する必要は必要ありません。
- 短期間(10画面以上/1人日)で開発が可能です。
- 段階的なシステム刷新(GUIへ移行)のプロセスを描けます。
- ファンクション、フォーム、WAMの全ての開発が行えます。

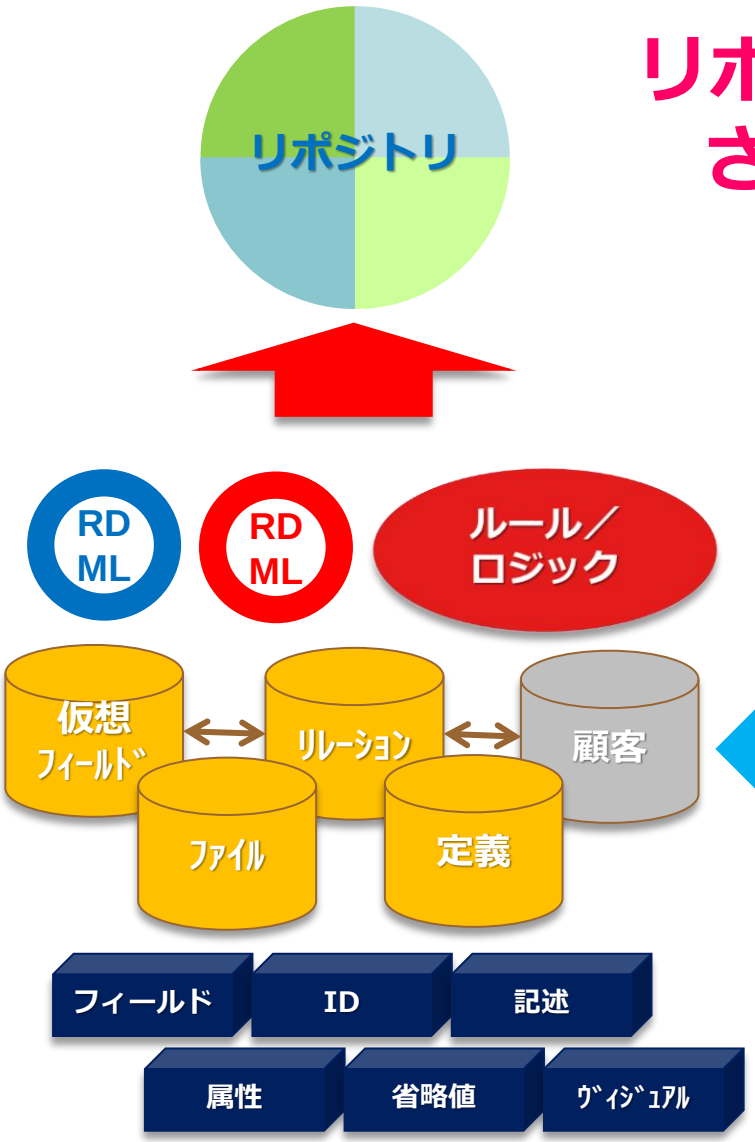
注意事項

- 複数の5250画面を1画面にすることはできません。
- LANSAでGUIを作成する技術が必要となります。
- JavaScriptの技術が必要とされる可能性が(大いに)あります。

2段階プロセス

- ステップ1 リポジトリを構築する
  - 開発またはアプリケーションのタイプに関わらず、まず最初にリポジトリを作成する必要があります。
  - 各アプリケーションは、同一のLANSAリポジトリを共有します。
  - 時間と労力を惜しまず、リポジトリを構築してください。
- ステップ2 RDMLロジックを構築する
  - RDMLを構築する前に、アプリケーションのスタイルを知っておく必要があります。
  - 各アプリケーションでは、共通言語のRDMLが使用されます。

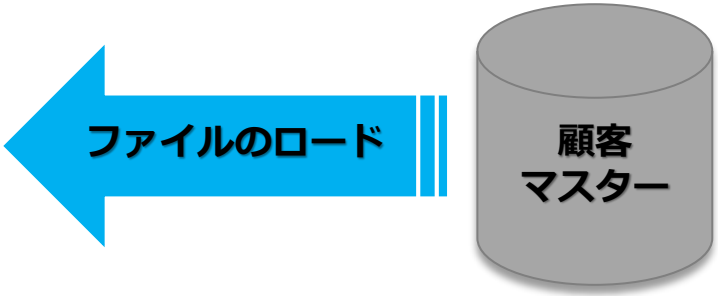
リポジトリの構築



リポジトリへの投資は共有され再利用されます!!!

① モデリング・ツールの使用

② パッケージ・アプリケーション
既存データベース・ファイル



③ マニュアルでの作成

RDMLの構築

アプリケーションに対してRDMLを構築するには、以下の
方法があります：

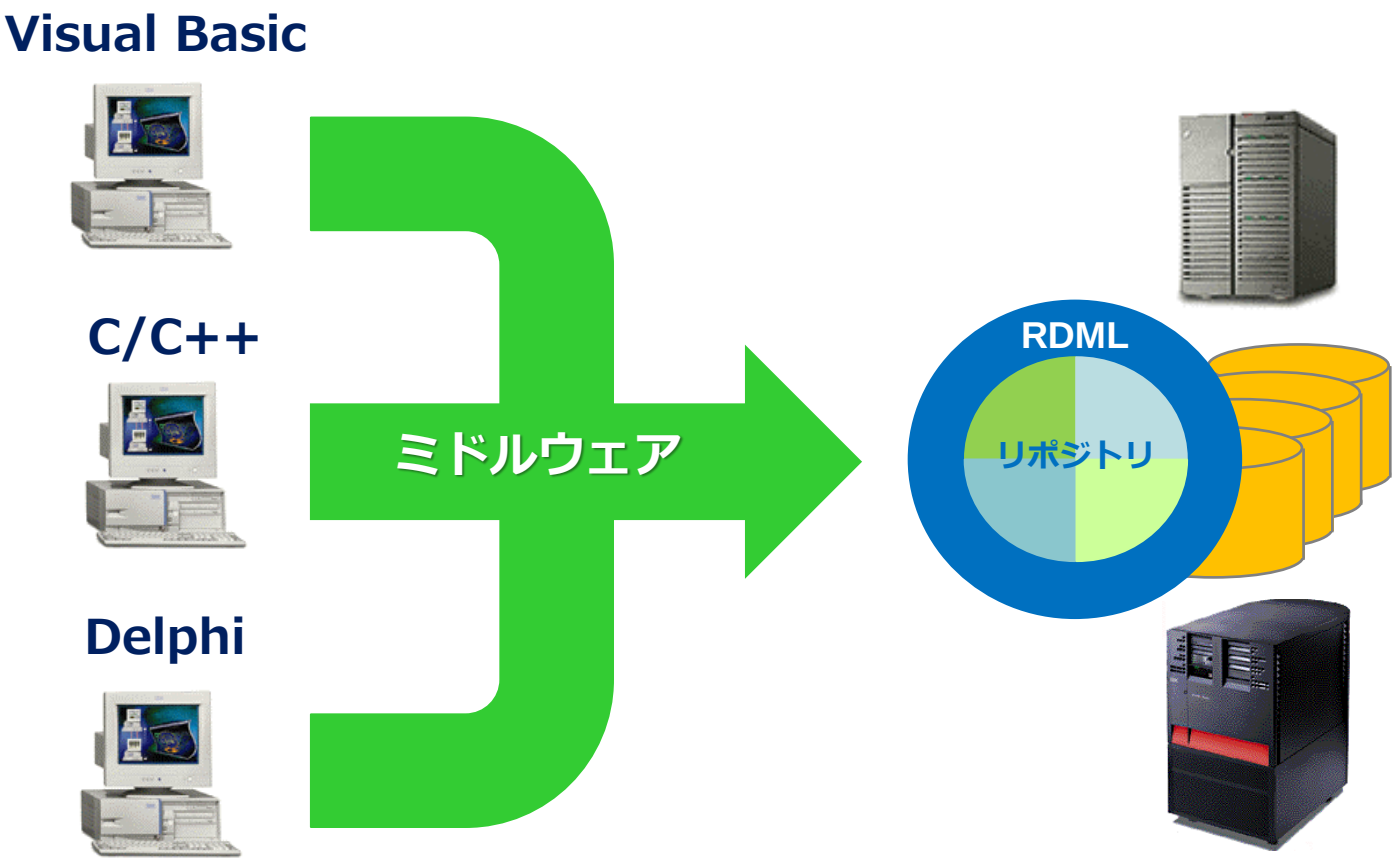
1. LANSAのプログラミング可能なテンプレートを使用する
2. マニュアルでRDMLをコーディングする

RDMLは、LANSA の IBM i 開発環境 および Windows 開発環境
を使用して開発できます。*

* Windows 最適化アプリケーションは、Windows で作成します。

LANSAのオープン・アーキテクチャ

RDML以外の言語からも、リポジトリを共有できます。

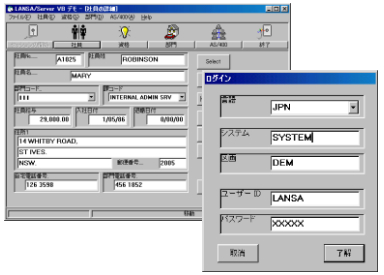


LANSA製品Family

LANSA

LANSA Integrator

LANSA Composer



LANSA Open

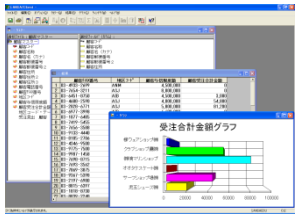
LongRange

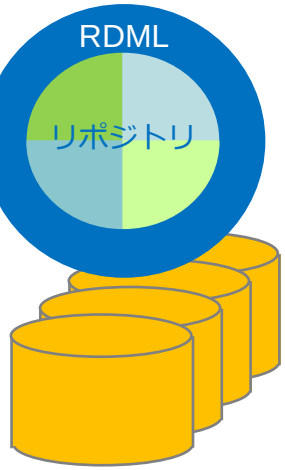
RDML

リポジトリ

aXes from LANSA

LANSA Client





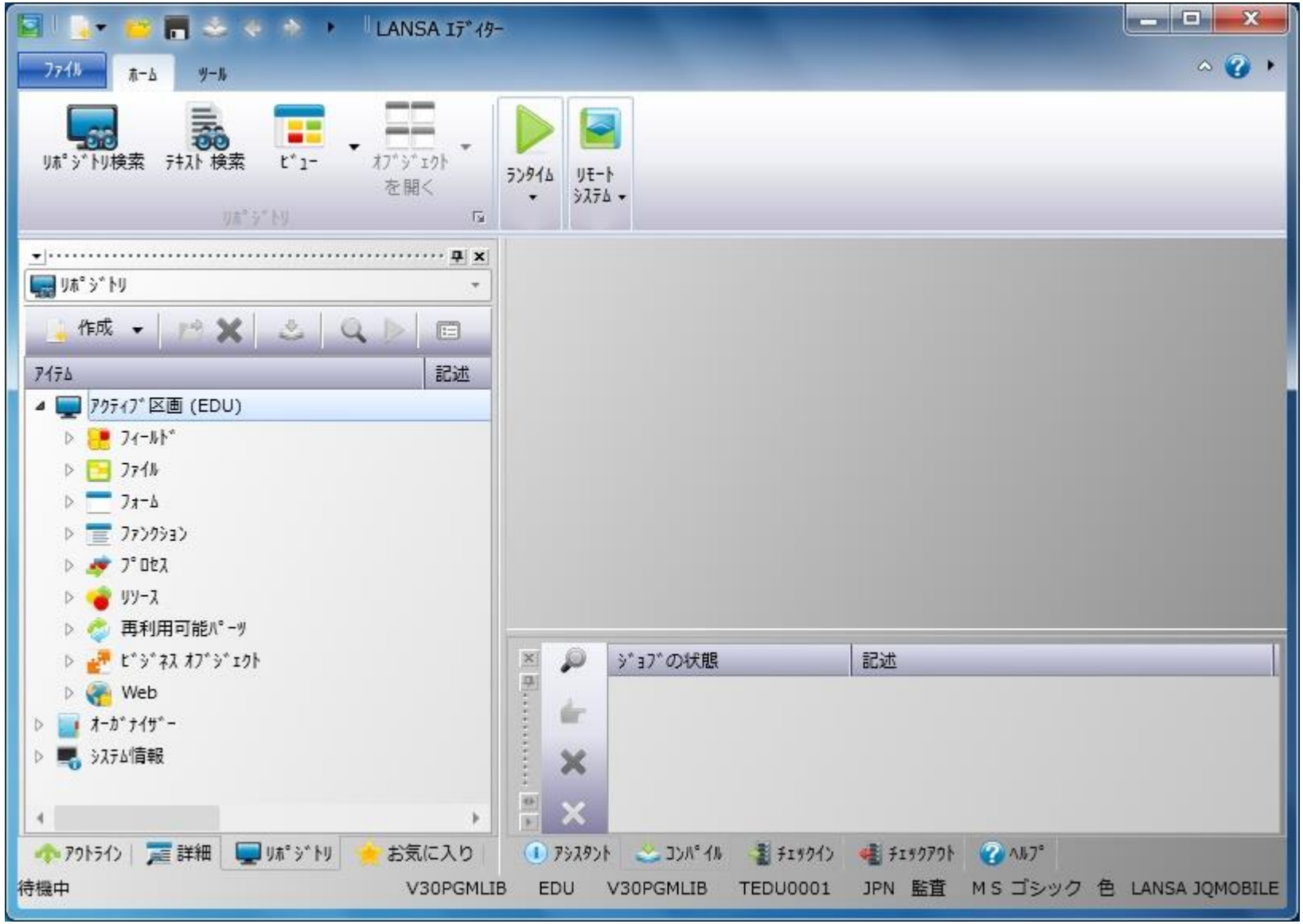
IDE(LANSA エディター)

IDEとは

IDE=Integrated Development Environment

- LANSAアプリケーションの開発、保守により良い方法：
  - 新しく、一貫した統合開発環境
  - 統合されたフィールドとファイル編集
  - 統合されたリポジトリブラウザ
    - LANSA Explorerを刷新
  - カストマイズ可能なレイアウトと機能
  - ソースエディターのワードラッピング機能
  - オート・コンプリート、コマンド・アシスタント、強化されたデバッギング機能による生産性向上

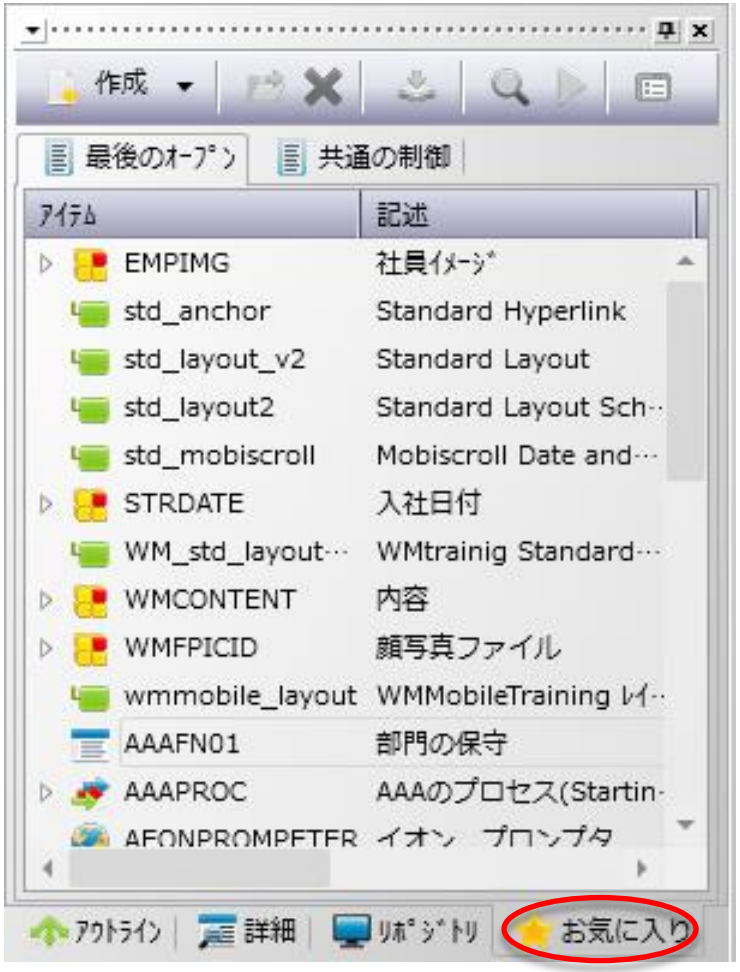
概観 – LANSA エディター



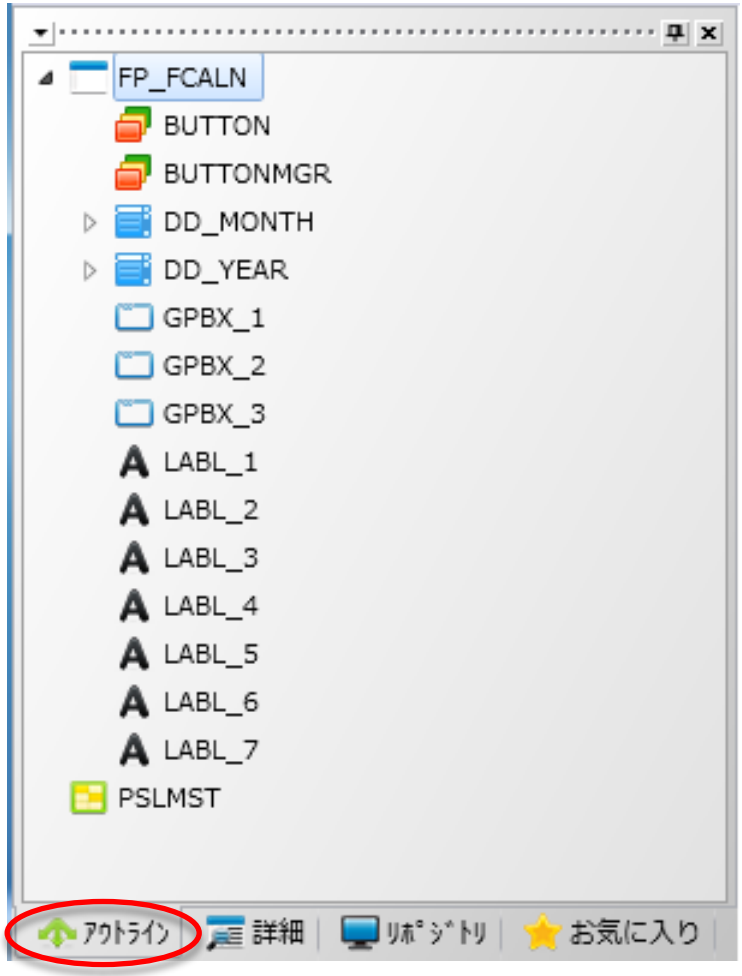
リポジトリ


リポジトリ
作成
アイテム
記述
アクティブ区画 (EDU)
フィールド
ファイル
フォーム
ファンクション
プロセス
リソース
再利用可能パーツ
ビジネス オブジェクト
Web
オーガナイザー
システム情報
アウトライン
詳細
リポジトリ
お気に入り

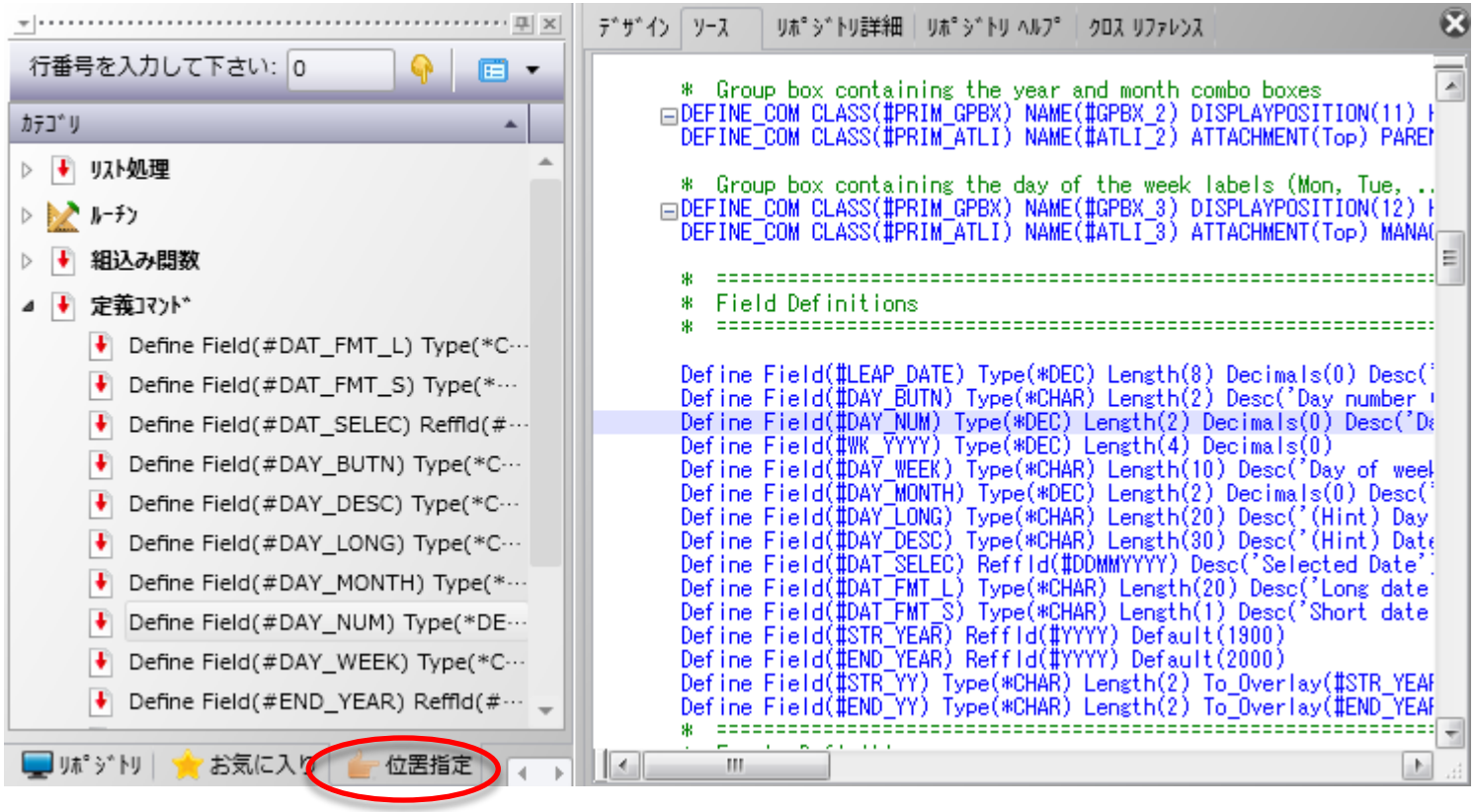
お気に入り


作成
最後のｵｰﾌﾟﾝ
共通の制御
ｱｲﾃﾑ
記述
EMPIMG
社員ｲﾒｰｼﾞ
std_anchor
Standard Hyperlink
std_layout_v2
Standard Layout
std_layout2
Standard Layout Sch
std_mobiscroll
Mobiscroll Date and
STRDATE
入社日付
WM_std_layout
WMtrainig Standard
WMCONTENT
内容
WMFPICID
顔写真ファイル
wmmobile_layout
WMMobileTraining ﾚｲ
AAAFN01
部門の保守
AAAPROC
AAAのプロセス(Startin
ｱｳﾄﾗｲﾝ
詳細
ﾘﾎﾟｼﾞﾄﾘ
お気に入り

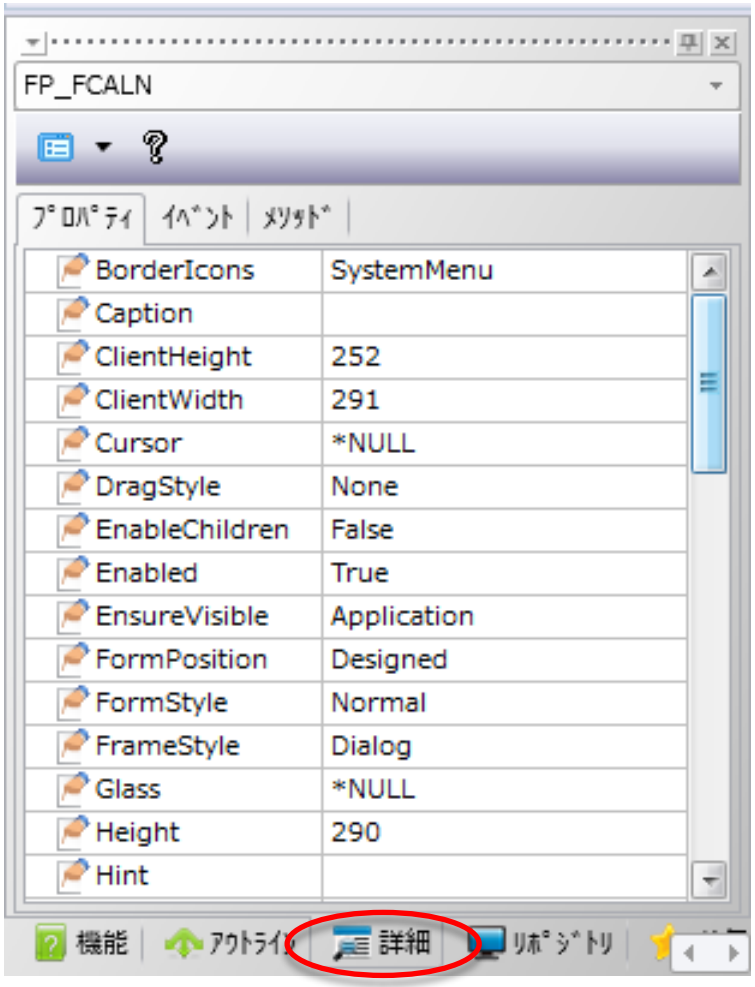
アウトライン


FP_FCALN
BUTTON
BUTTONMGR
DD_MONTH
DD_YEAR
GPBX_1
GPBX_2
GPBX_3
LABL_1
LABL_2
LABL_3
LABL_4
LABL_5
LABL_6
LABL_7
PSLMST
ｱｳﾄﾗｲﾝ
詳細
ﾘﾎﾟｼﾞﾄﾘ
お気に入り

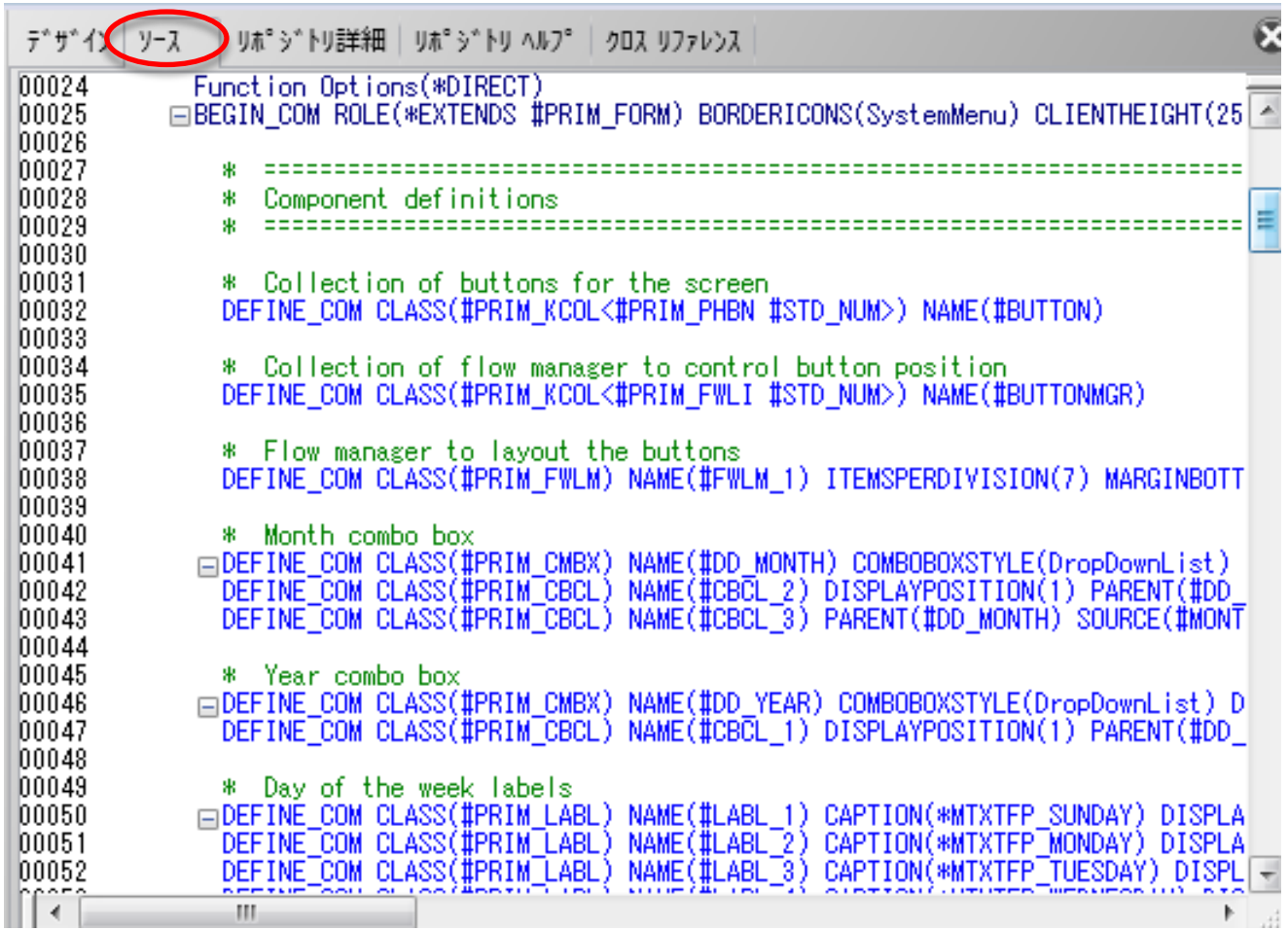
位置指定


行番号を入力して下さい: 0
ｶﾃｺﾞﾘ
ﾘｽﾄ処理
ﾙｰﾁﾝ
組込み関数
定義ｺﾏﾝﾄﾞ
Define Field(#DAT_FMT_L) Type(*C…
Define Field(#DAT_FMT_S) Type(*…
Define Field(#DAT_SELEC) Reffld(#…
Define Field(#DAY_BUTN) Type(*C…
Define Field(#DAY_DESC) Type(*C…
Define Field(#DAY_LONG) Type(*C…
Define Field(#DAY_MONTH) Type(*…
Define Field(#DAY_NUM) Type(*DE…
Define Field(#DAY_WEEK) Type(*C…
Define Field(#END_YEAR) Reffld(#…
ﾘﾎﾟｼﾞﾄﾘ
お気に入り
位置指定
ﾃﾞｻﾞｲﾝ
ｿｰｽ
ﾘﾎﾟｼﾞﾄﾘ詳細
ﾘﾎﾟｼﾞﾄﾘ ﾍﾙﾌﾟ
ｸﾛｽ ﾘﾌｧﾚﾝｽ
* Group box containing the year and month combo boxes
DEFINE_COM CLASS(#PRIM_GPBX) NAME(#GPBX_2) DISPLAYPOSITION(11)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_2) ATTACHMENT(Top) PAREN
* Group box containing the day of the week labels (Mon, Tue, ..
DEFINE_COM CLASS(#PRIM_GPBX) NAME(#GPBX_3) DISPLAYPOSITION(12)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_3) ATTACHMENT(Top) MANA
* ==========
* Field Definitions
* ==========
Define Field(#LEAP_DATE) Type(*DEC) Length(8) Decimals(0) Desc('
Define Field(#DAY_BUTN) Type(*CHAR) Length(2) Desc('Day number
Define Field(#DAY_NUM) Type(*DEC) Length(2) Decimals(0) Desc('Da
Define Field(#WK_YYYY) Type(*DEC) Length(4) Decimals(0)
Define Field(#DAY_WEEK) Type(*CHAR) Length(10) Desc('Day of week
Define Field(#DAY_MONTH) Type(*DEC) Length(2) Decimals(0) Desc('
Define Field(#DAY_LONG) Type(*CHAR) Length(20) Desc('(Hint) Day
Define Field(#DAY_DESC) Type(*CHAR) Length(30) Desc('(Hint) Date
Define Field(#DAT_SELEC) Reffld(#DDMMYYYY) Desc('Selected Date'
Define Field(#DAT_FMT_L) Type(*CHAR) Length(20) Desc('Long date
Define Field(#DAT_FMT_S) Type(*CHAR) Length(1) Desc('Short date
Define Field(#STR_YEAR) Reffld(#YYYY) Default(1900)
Define Field(#END_YEAR) Reffld(#YYYY) Default(2000)
Define Field(#STR_YY) Type(*CHAR) Length(2) To_Overlay(#STR_YEAR
Define Field(#END_YY) Type(*CHAR) Length(2) To_Overlay(#END_YEAR
* ==========

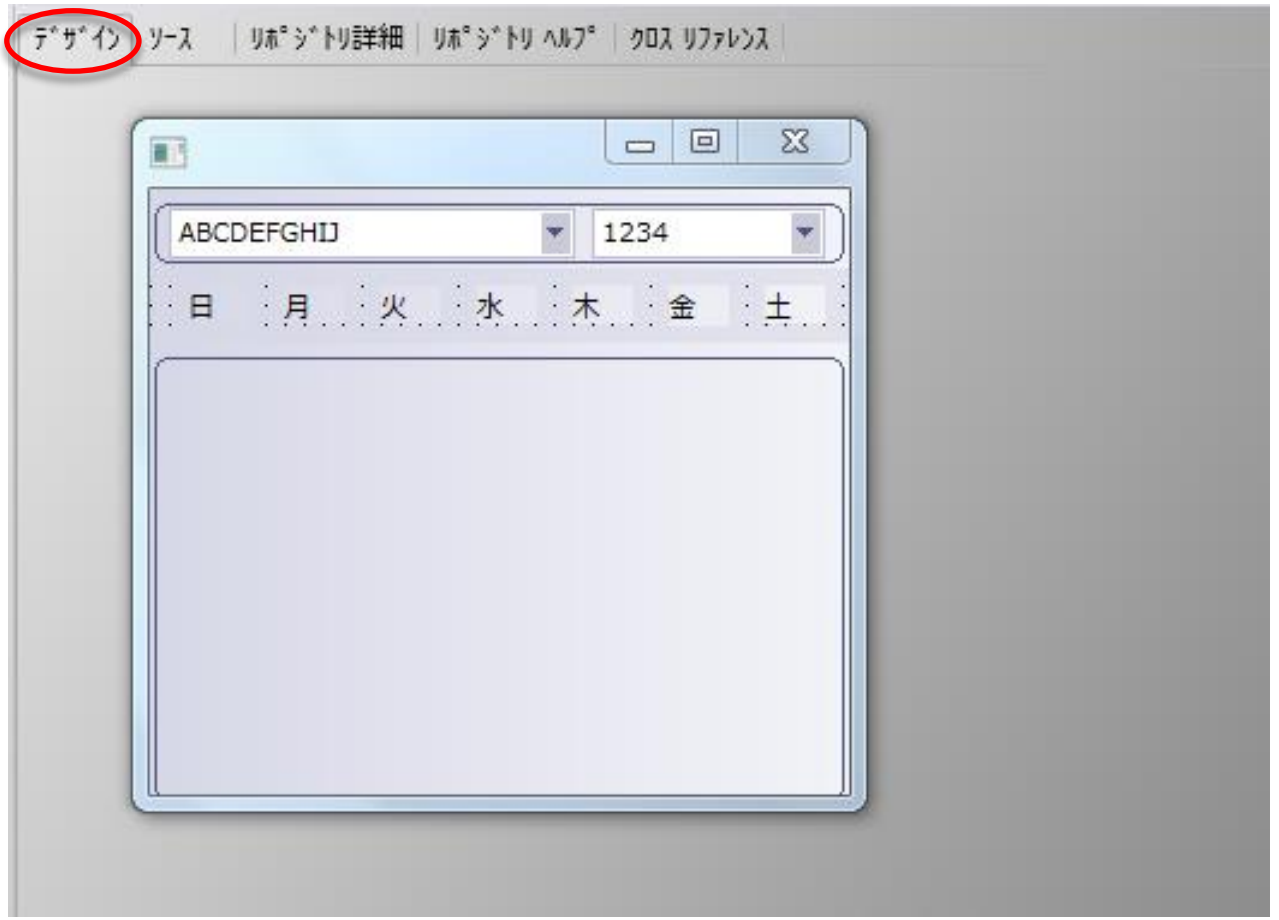
詳細


FP_FCALN
ﾌﾟﾛﾊﾟﾃｨ
ｲﾍﾞﾝﾄ
ﾒｿｯﾄﾞ
BorderIcons
SystemMenu
Caption
ClientHeight
252
ClientWidth
291
Cursor
*NULL
DragStyle
None
EnableChildren
False
Enabled
True
EnsureVisible
Application
FormPosition
Designed
FormStyle
Normal
FrameStyle
Dialog
Glass
*NULL
Height
290
Hint
機能
詳細
ﾘﾎﾟｼﾞﾄﾘ

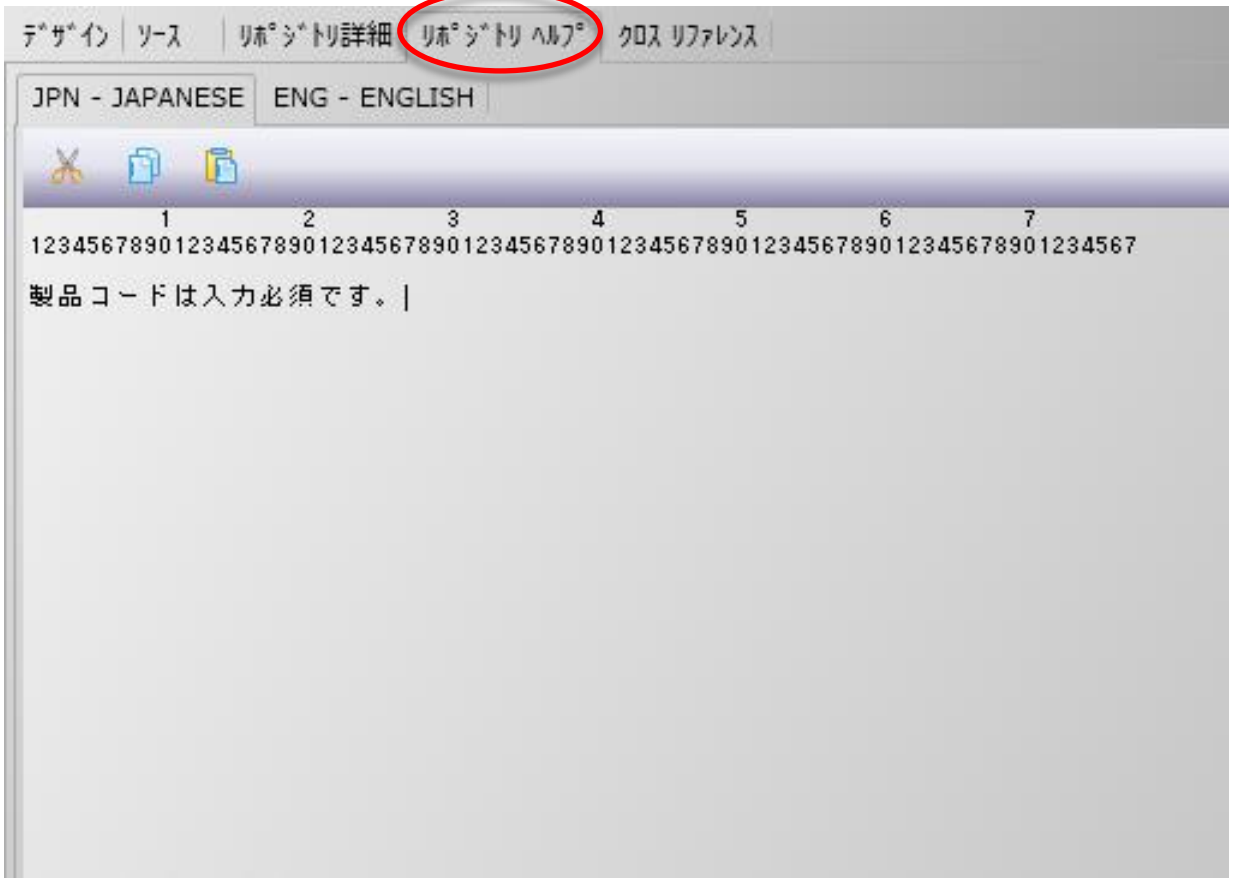
ソース


デザイン
ソース
リポジトリ詳細
リポジトリ ヘルプ
クロス リファレンス
00024 Function Options(*DIRECT)
00025 BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) BORDERICONS(SystemMenu) CLIENTHEIGHT(25
00026
00027 * ==========
00028 * Component definitions
00029 * ==========
00030
00031 * Collection of buttons for the screen
00032 DEFINE_COM CLASS(#PRIM_KCOL<#PRIM_PHBN #STD_NUM>) NAME(#BUTTON)
00033
00034 * Collection of flow manager to control button position
00035 DEFINE_COM CLASS(#PRIM_KCOL<#PRIM_FWLI #STD_NUM>) NAME(#BUTTONMGR)
00036
00037 * Flow manager to layout the buttons
00038 DEFINE_COM CLASS(#PRIM_FWLM) NAME(#FWLM_1) ITEMSPERDIVISION(7) MARGINBOTT
00039
00040 * Month combo box
00041 DEFINE_COM CLASS(#PRIM_CMBX) NAME(#DD_MONTH) COMBOBOXSTYLE(DropDownList)
00042 DEFINE_COM CLASS(#PRIM_CBCL) NAME(#CBCL_2) DISPLAYPOSITION(1) PARENT(#DD_
00043 DEFINE_COM CLASS(#PRIM_CBCL) NAME(#CBCL_3) PARENT(#DD_MONTH) SOURCE(#MONT
00044
00045 * Year combo box
00046 DEFINE_COM CLASS(#PRIM_CMBX) NAME(#DD_YEAR) COMBOBOXSTYLE(DropDownList) D
00047 DEFINE_COM CLASS(#PRIM_CBCL) NAME(#CBCL_1) DISPLAYPOSITION(1) PARENT(#DD_
00048
00049 * Day of the week labels
00050 DEFINE_COM CLASS(#PRIM_LABL) NAME(#LABL_1) CAPTION(*MTXTFP_SUNDAY) DISPLA
00051 DEFINE_COM CLASS(#PRIM_LABL) NAME(#LABL_2) CAPTION(*MTXTFP_MONDAY) DISPLA
00052 DEFINE_COM CLASS(#PRIM_LABL) NAME(#LABL_3) CAPTION(*MTXTFP_TUESDAY) DISPL

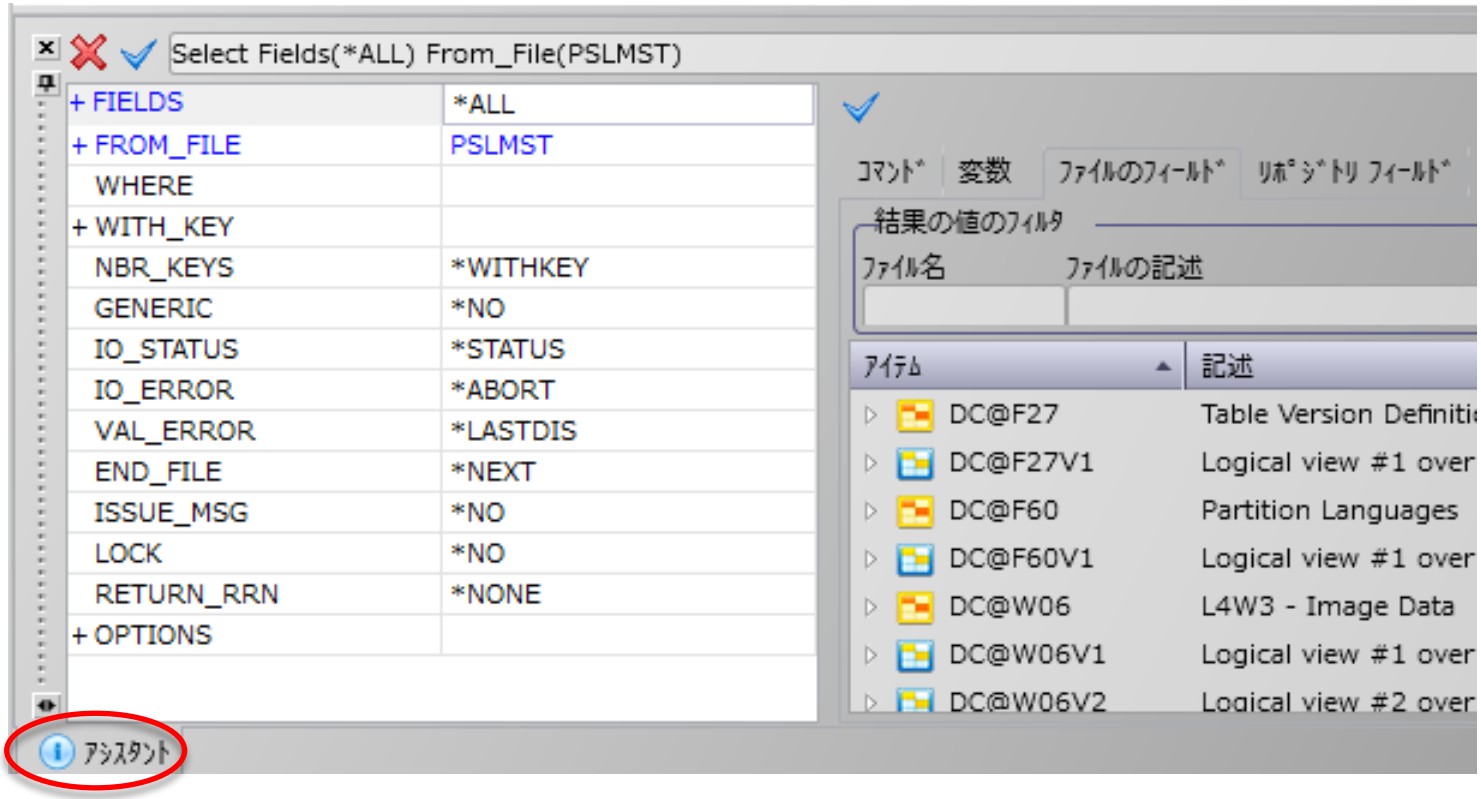
デザイン


ﾃﾞｻﾞｲﾝ
ｿｰｽ
ﾘﾎﾟｼﾞﾄﾘ詳細
ﾘﾎﾟｼﾞﾄﾘ ﾍﾙﾌﾟ
ｸﾛｽ ﾘﾌｧﾚﾝｽ
ABCDEFGHIJ
1234
日
月
火
水
木
金
土

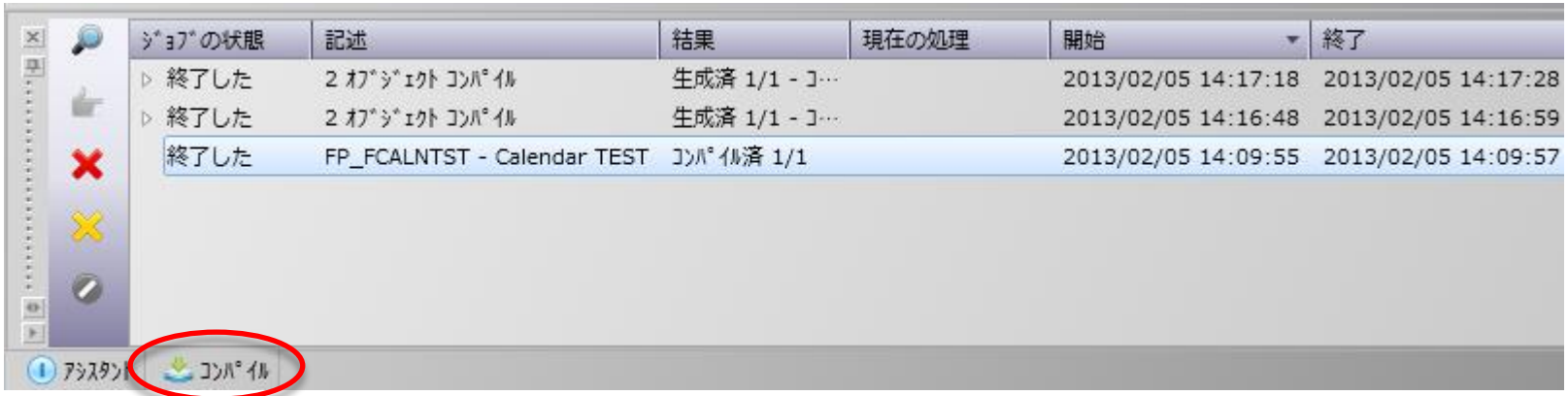
リポジトリヘルプ


ﾃﾞｻﾞｲﾝ
ｿｰｽ
ﾘﾎﾟｼﾞﾄﾘ詳細
ﾘﾎﾟｼﾞﾄﾘ ﾍﾙﾌﾟ
ｸﾛｽ ﾘﾌｧﾚﾝｽ
JPN - JAPANESE
ENG - ENGLISH
1 2 3 4 5 6 7
12345678901234567890123456789012345678901234567890123456789012345678901234567
製品コードは入力必須です。

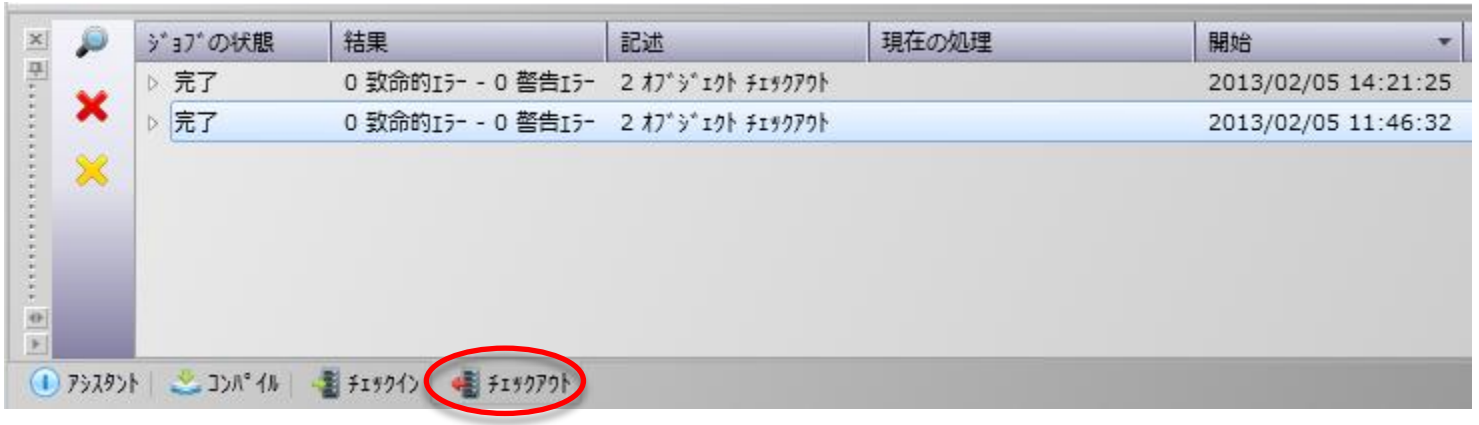
アシスタント


Select Fields(*ALL) From_File(PSLMST)
+ FIELDS
*ALL
+ FROM_FILE
PSLMST
WHERE
+ WITH_KEY
NBR_KEYS
*WITHKEY
GENERIC
*NO
IO_STATUS
*STATUS
IO_ERROR
*ABORT
VAL_ERROR
*LASTDIS
END_FILE
*NEXT
ISSUE_MSG
*NO
LOCK
*NO
RETURN_RRN
*NONE
+ OPTIONS
ｺﾏﾝﾄﾞ
変数
ﾌｧｲﾙのﾌｨｰﾙﾄﾞ
ﾘﾎﾟｼﾞﾄﾘ ﾌｨｰﾙﾄﾞ
結果の値のﾌｨﾙﾀ
ﾌｧｲﾙ名
ﾌｧｲﾙの記述
ｱｲﾃﾑ
記述
DC@F27
Table Version Definiti
DC@F27V1
Logical view #1 over
DC@F60
Partition Languages
DC@F60V1
Logical view #1 over
DC@W06
L4W3 - Image Data
DC@W06V1
Logical view #1 over
DC@W06V2
Logical view #2 over
ｱｼｽﾀﾝﾄ

コンパイル


ｼﾞｮﾌﾞの状態
記述
結果
現在の処理
開始
終了
終了した
2 ｵﾌﾞｼﾞｪｸﾄ ｺﾝﾊﾟｲﾙ
生成済 1/1 - ｺ…
2013/02/05 14:17:18
2013/02/05 14:17:28
終了した
2 ｵﾌﾞｼﾞｪｸﾄ ｺﾝﾊﾟｲﾙ
生成済 1/1 - ｺ…
2013/02/05 14:16:48
2013/02/05 14:16:59
終了した
FP_FCALNTST - Calendar TEST
ｺﾝﾊﾟｲﾙ済 1/1
2013/02/05 14:09:55
2013/02/05 14:09:57
ｱｼｽﾀﾝﾄ
ｺﾝﾊﾟｲﾙ

チェックイン/チェックアウト



リポジトリ基礎

詳細な内容は「LANSA Repository」コースで取り扱います。

リポジトリとは

フィールド列・並び順・固有なキー
書込・削除条件

型・長さ・名称・許される値

各オブジェクトの履歴

参照・発生条件

プログラムのふるまい
画面のふるまい

リポジトリとは何か

- リポジトリは、アプリケーションについての情報を１ケ所にまとめて保管し、一元管理します。
- フォーマットはプラットフォームに依存しません。
- アプリケーション・コーティングの作業量を減らします。
- アプリケーション・ロジックの 共有 と 再利用 を促進します。
- アプリケーションの保守が簡素化されます。
- 開発担当者の生産性が向上します。

しかし 全てのリポジトリが同じものであると
言うことではありません..

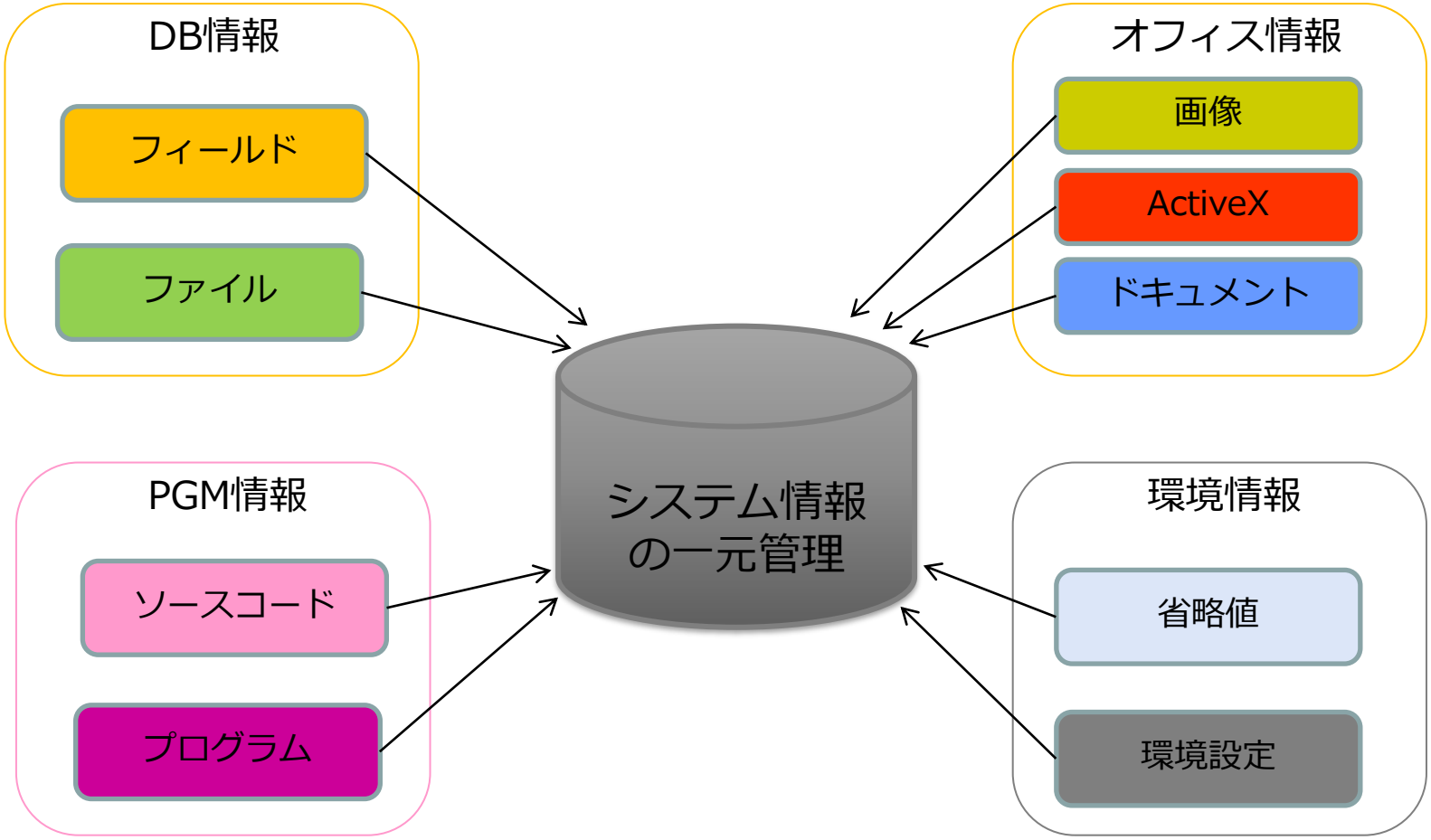
アクティブ・リポジトリ

- 多くの製品に“リポジトリ”がありますが、それはプログラムの構築にのみ使用されるものです
- アクティブ リポジトリは、アプリケーションの構築と実行に使用されます。ダイナミックであり、状況変化に対応します。
- したがって…

  アプリケーション・プログラムを再構築しなくても、変更を行うことが可能です。

LANSAはアクティブ・リポジトリ・アーキテクチャです。

LANSAリポジトリに保管されるもの

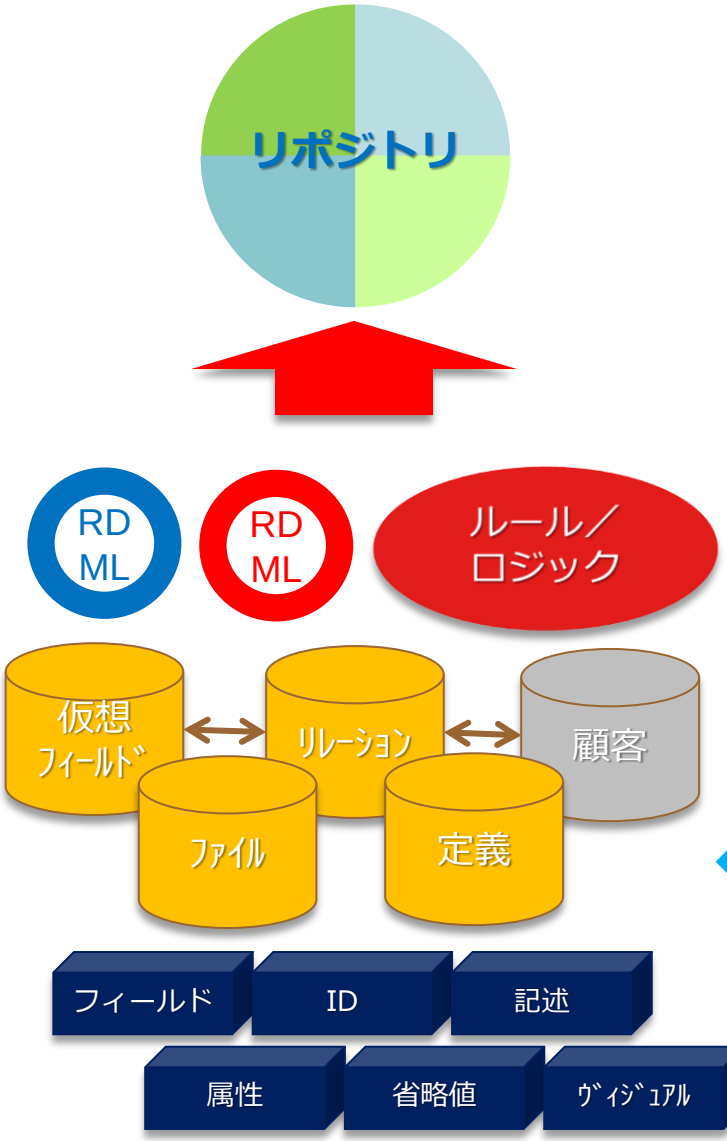


システムに必要なあらゆる情報を一箇所に保管します

リポジトリの定義

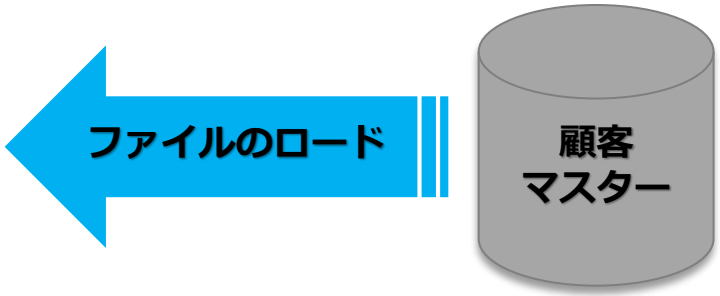
- フィールド定義
  - （記述、ラベル、欄見出し、I/O属性、省略値、ヘルプ・テキスト、プロンプト機能、トリガー、複数言語など）
- ファイル定義
  - （物理ファイル／テーブル、論理ファイル／ビュー、ファイル・リレーション、トリガーなど）
- ルール
  - （範囲検査、リスト検査、簡単なロジック、複雑なロジック、ファイル検査、日付形式／検査）
- コンポーネント*
  - （プロパティ、メソッド、イベント、ルーティン、イメージ、リスト、グラフ、ボタンなど）
- 特殊機能
  - （複数言語対応、仮想フィールド、事前定義結合フィールド、バッチ制御など）

既存のアプリケーションを使用する

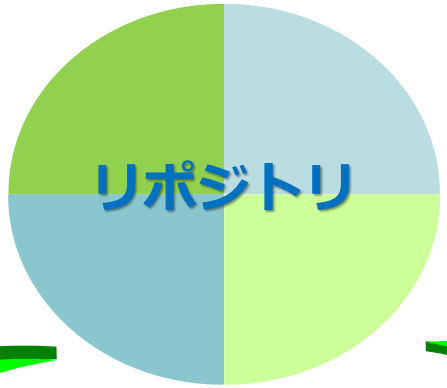


LANSA を使用して、
既存のアプリケーションと
インテグレートする
ことができます

既存アプリケーション
既存のデータベース・ファイル



リポジトリのインプリメント



LANSA リポジトリ・データベース

フィールド
ヘルプテキスト
複数言語
アクセス経路

オブジェクトアクセスモジュール

妥当性検査
仮想フィールド
トリガー, etc.

オブジェクト・アクセス・モジュール

LANSA

OAM は

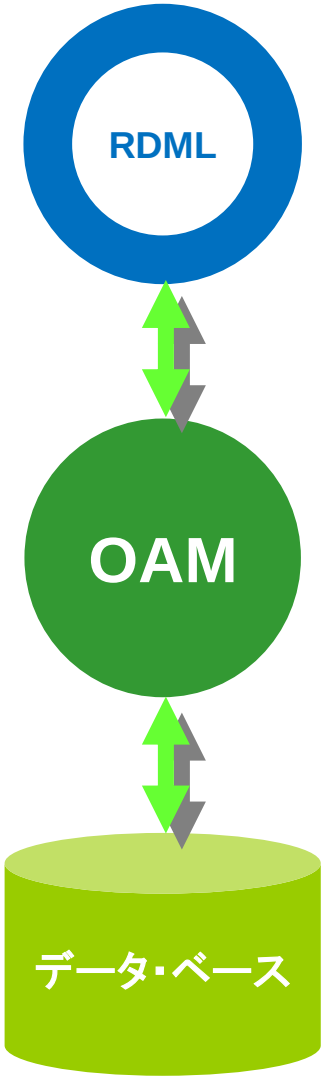
- 各ファイルに1つずつ作成される
- 全てのファイル・リクエストを処理する
- コンパイル済みプログラムである

そのメリット

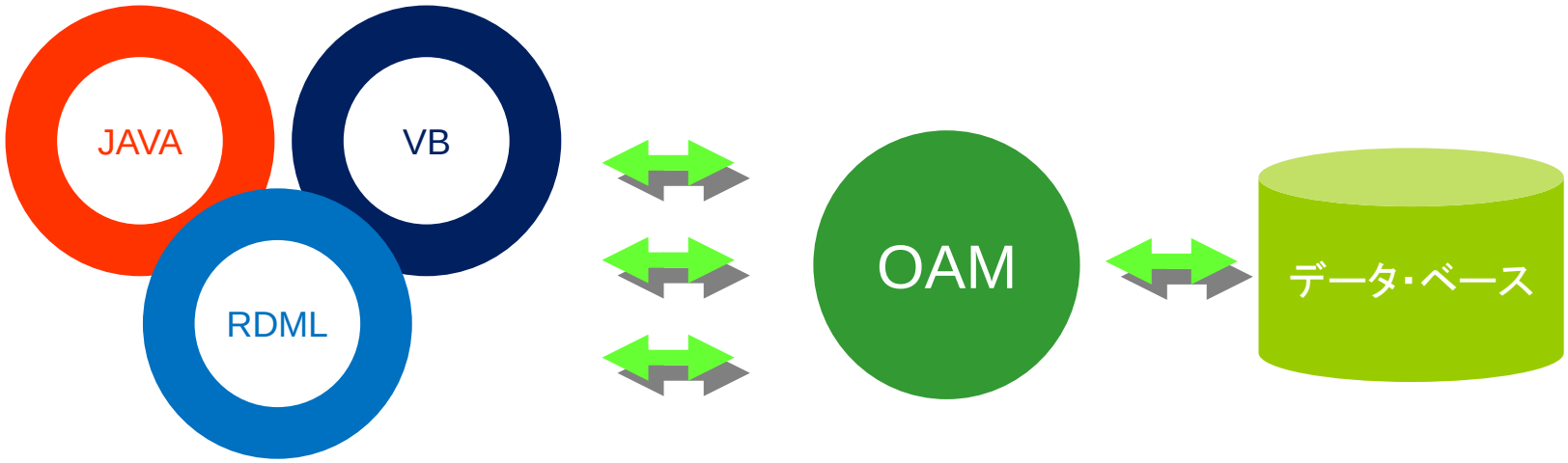
- プログラムを変更せずに、ルールを変更できる
- データから独立している

知っていて欲しいこと

- 特別な LANSA の機能
- 既存のデータベースを取り込む事ができる。

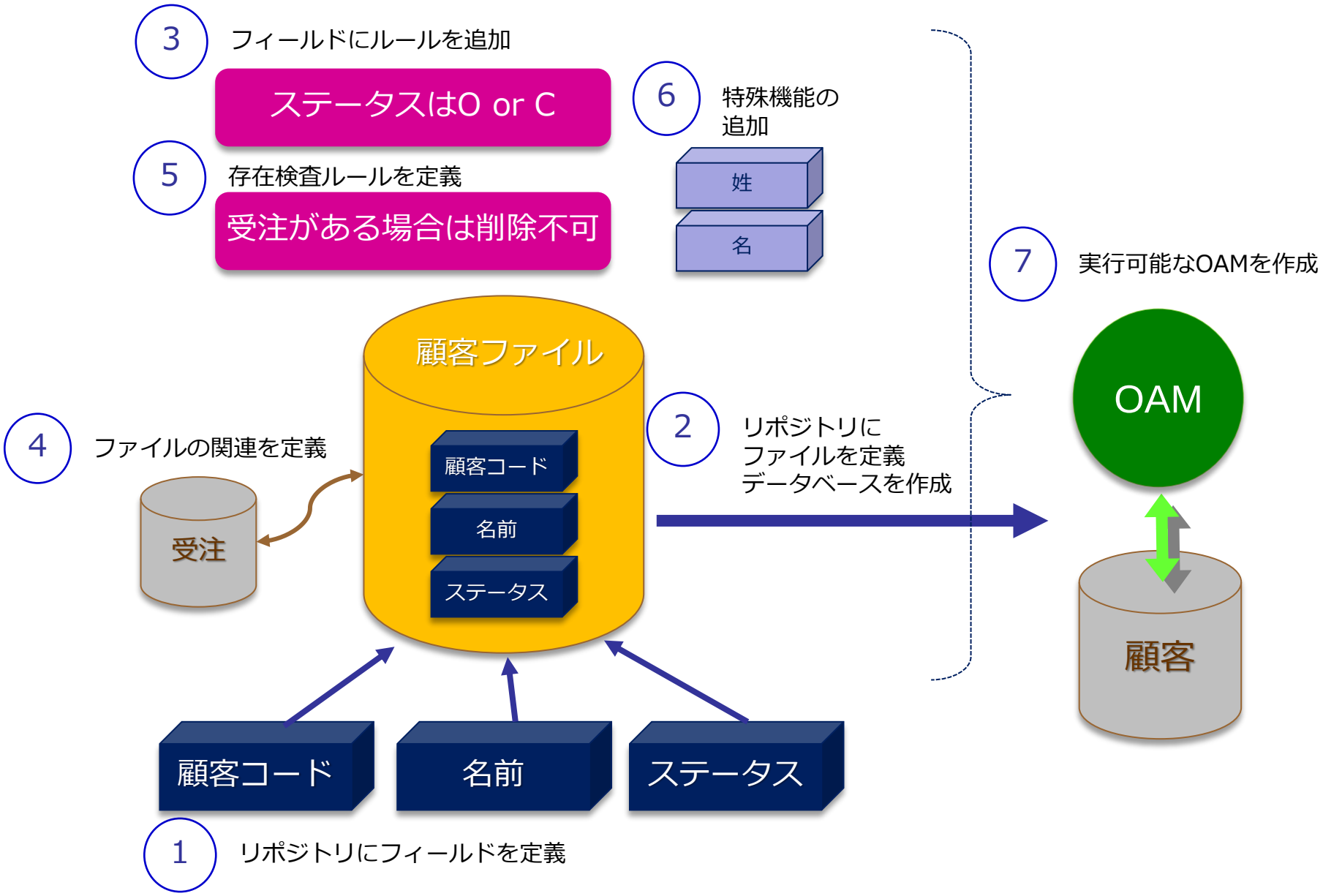


アプリケーションロジックとOAM



- OAM は全てのファイルに対して妥当性検査を実施し、アプリケーションロジックにメッセージを渡します。
- ルールは変更可能なので、プログラム自体を変更しなくても新しい OAM を作成できます。
- このファンクションをクライアント側に定義することもできます。Visual Basic やCで書かれたプログラムであっても構いません。

リポジトリを使用したファイルの開発

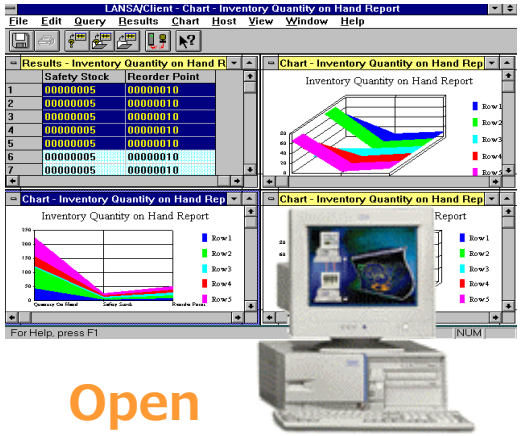


LANSAリポジトリを使用する

Windows

Web

XML



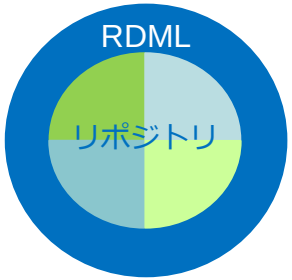
Open

AS/400

OAM

顧客

Status ='O' or 'C' or 'H'



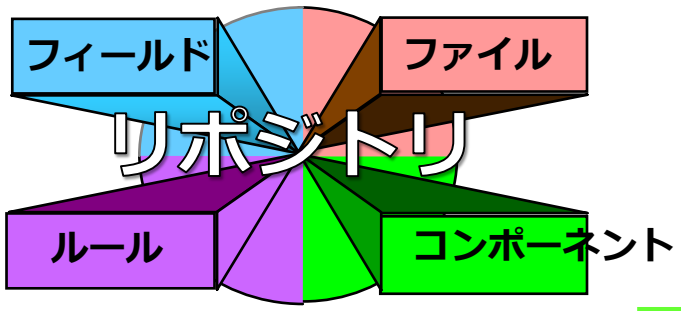
新しいビジネス・ルールを追加して、
新しいOAMを構築します。

まとめ

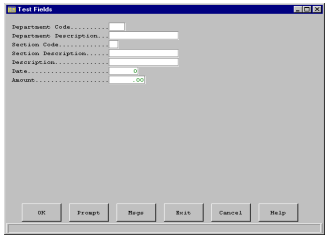
- 情報を一元管理し、ロジックを（ホストおよびクライアント／サーバーで）共有し再使用できるようにします。
- 開発担当者の生産性とアプリケーションのパフォーマンスを向上させます（シン・クライアント）。
- 定義を一元管理するため、アプリケーションの保守が簡素化されます。
- 強力な開発ツールがあります（仮想フィールドやPJFなど）。
- 既存データベースを処理することができます。
- ツールを自由に選択できます。
- 他のプラットフォーム上に存在することが可能です。

a. フィールド定義

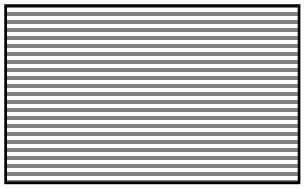
フィールド定義はどのように使用されるのか？



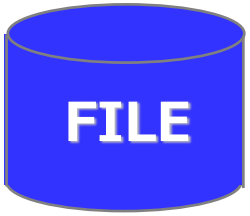
使用される



画面



レポート



ファイル

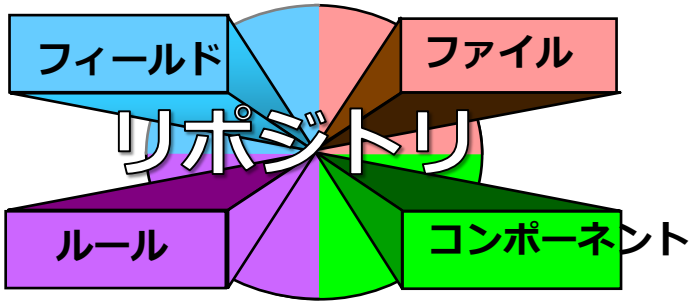
プログラム

各フィールドの明細を
注意深く確認しましょう！

フィールドのネーミング標準

フィールドのフォーマット

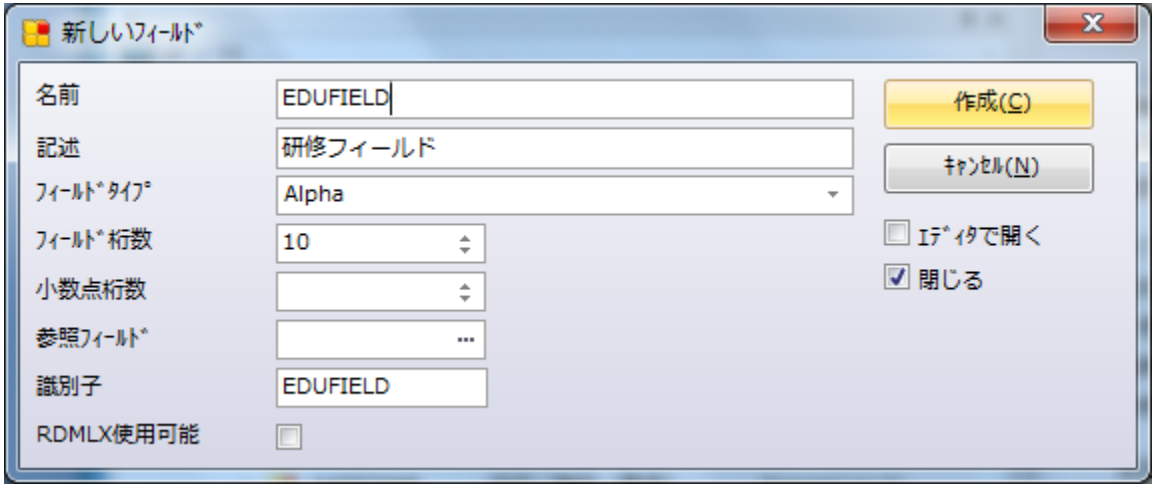
xxxxxxiii



- 最大で 9 文字
- # や @ は使用しない
- 理解しやすい名前を使用する
- 作業フィールドあるいは一時フィールドには iii を使用する
- リポジトリ内に全てのフィールドを定義する

新しいフィールドを作成する

([ファイル]) - [作成] – [フィールド]



フィールド定義の明細

LANSA

タイプ
(RDML)
- Alpha
- Decimal
- Signed

(RDMLX)
- Alpha
- Decimal
- Signed
- Char
- Binary
- Date
- Time
- Date Time
- Integer
- Flota
- String
- VarBinary
- CLOB
- BLOB
- NChar
- NVarVhar
- Boolean

プロンプト
- プロセス／ファンクションのプロンプトが可能

省略値
- RDMLによって使用される

記述（各言語）
- 記述
- ラベル
- 欄見出し

編集オプション
- 編集コード
- 編集語

鍵盤シフト
- O (半角+全角)
- J (全角のみ)
- W (半角カナ)

入力／出力属性

共通入力属性：

- (RDML) LC、ME、FE、RB
- (RDMLX) ASQN、ISO、SUTC、SUNS

出力属性：スタンプ属性

- USRX、JNMX、TIMU、DATC 等

グラフィック属性 ＊：

- PBnn、RBnn、CBOX、DDxx

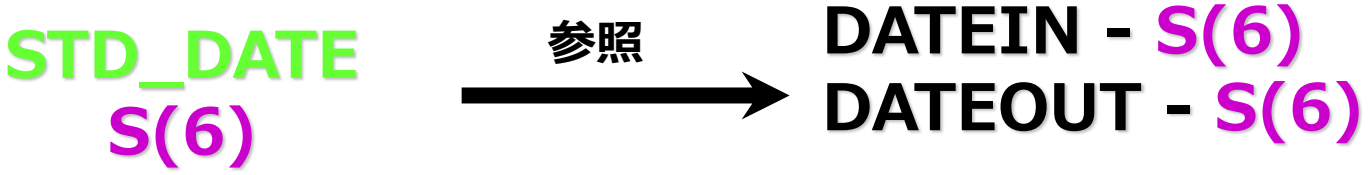
＊ 注：フィールドのグラフィック属性は、
グラフィック・コンポーネントと同じではありません。
プッシュボタン・コンポーネントは、LANSA コンポーネント・エディタを使用して作成されます。

参照フィールド

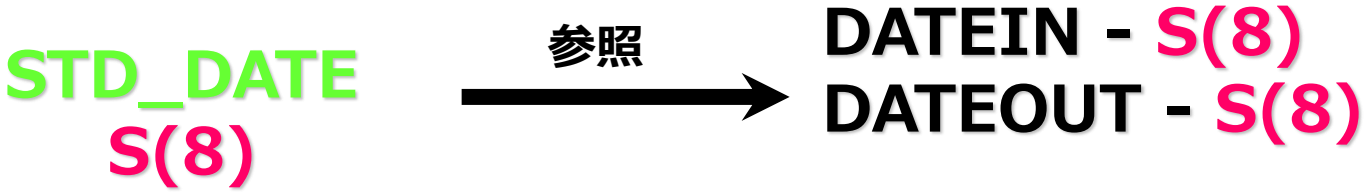
- フィールド明細の継承を促進します
  （タイプ、長さ、小数点、編集オプション、省略値、
  入力 & 出力属性）

- 参照元のフィールドに変更が加えられると、
  参照先のフィールドも自動的にアップデートされます。

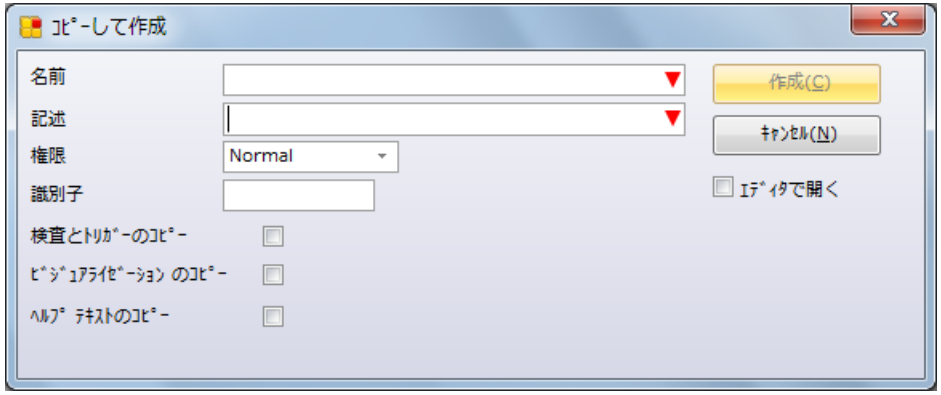
- ヘルプ・テキスト、ルール／トリガーをコピーできます。



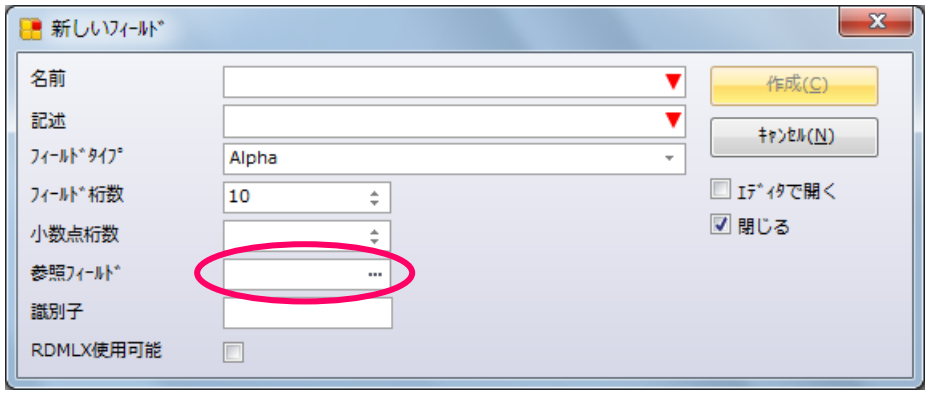
こう変更されると：



コピーと参照フィールドの違い



コピーは１回限りの継承です。



参照は、永久的なリンクです。

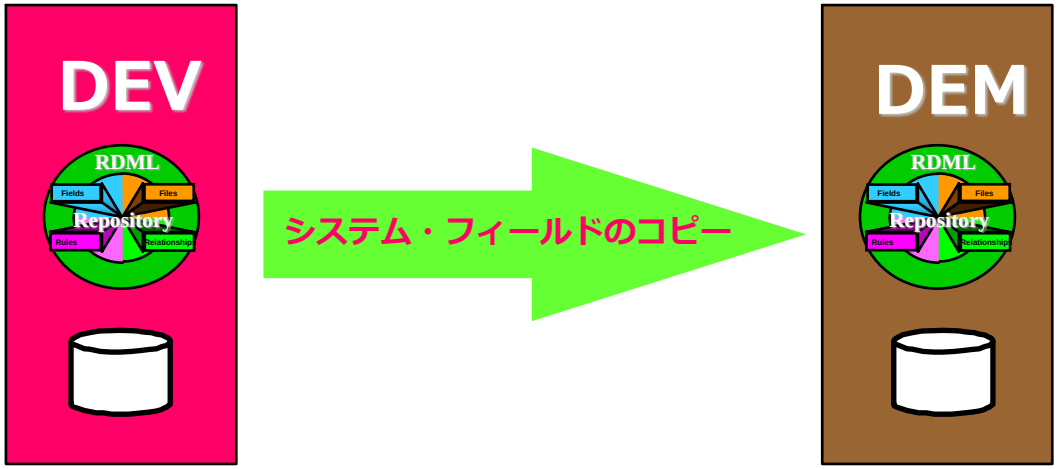
コピーにより、フィールドを作成する場合、
「妥当性検査」、「ビジュアライゼーション」、
「ヘルプテキスト」のコピーが選択できます。

システム・フィールド

システム・フィールド：

- どのフィールドもシステム・フィールドとして指定することが可能
- 削除不可（NO に変更された場合は可能）
- 新しい区画の作成時にコピー可能
- STD_xxxxx フィールドとともに使用される

システム・フィールド “YES” の場合：



演習1:フィールド作成

b. ファイル定義

フィールド/レコード/ファイル/テーブル

LANSA

フィールド

名前	ID番号	住所

レコード

J.Smith	645674	6 King St …

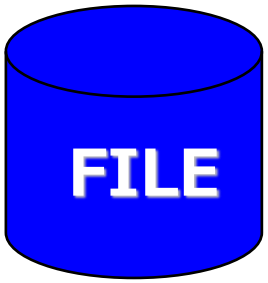
ファイル

名前	ID 番号	住所
J.Smith	645674	6 King St …
M.Brown	349534	83 Pitt St…
etc.	etc.	etc.

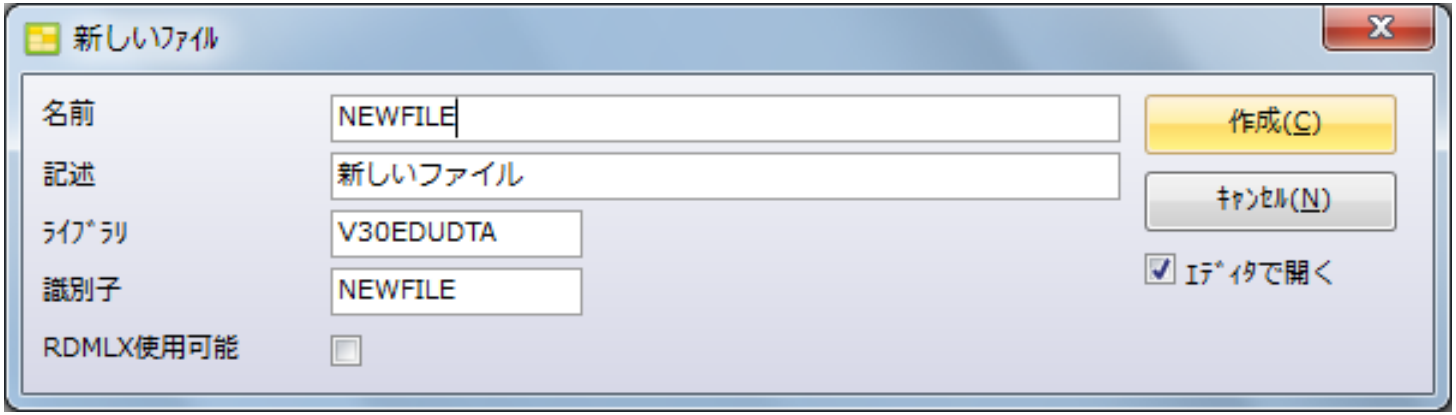
物理ファイル
論理ファイル

同じ

テーブル
インデックス／ビュー



ファイル定義を作成する



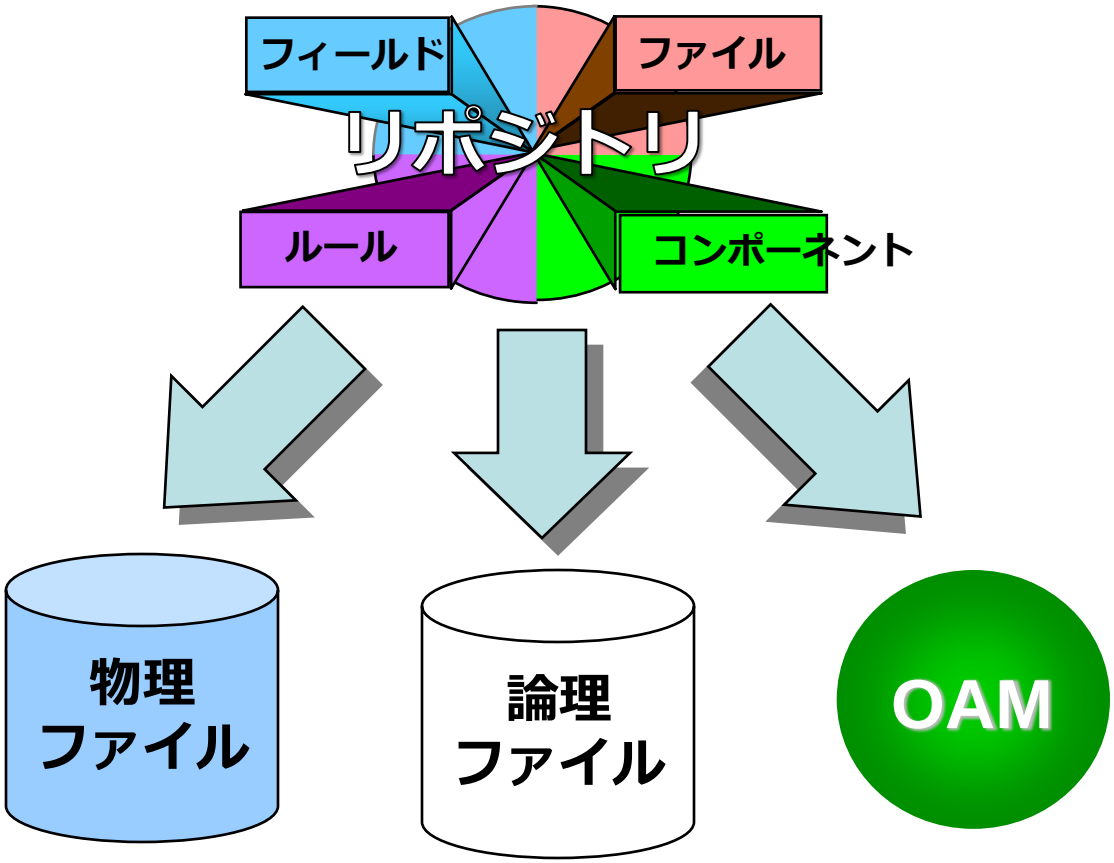
ファイルは以下の方法により作成することができます。

- マニュアルによる作成　（新規作成）
- 既存ファイルのコピー
- 外部ファイルのロード
- モデリング・ツールを使用

ファイルにフィールドを追加する

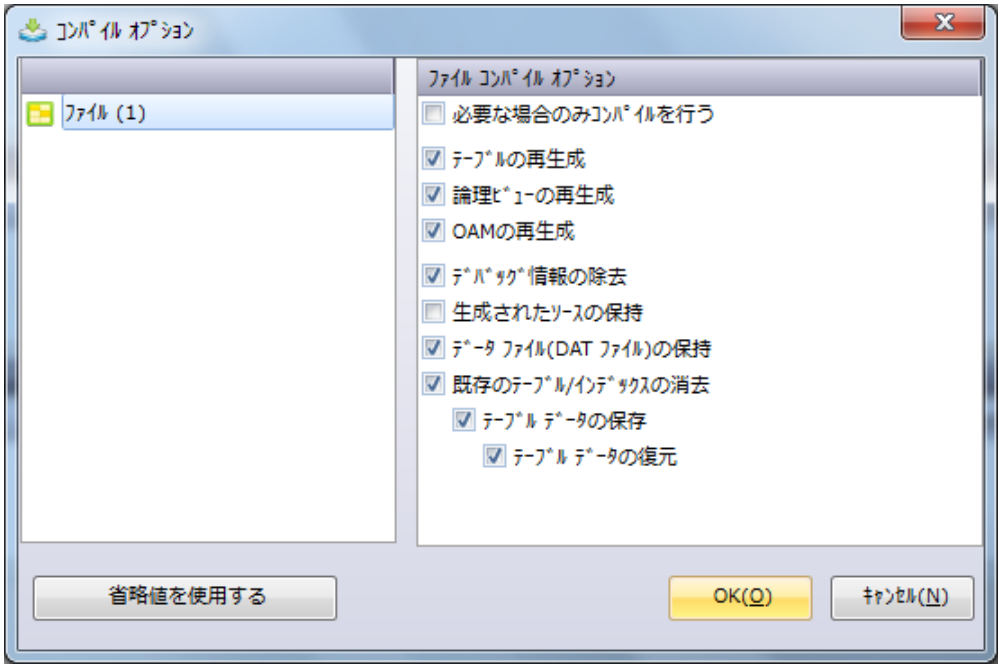
「順序」とは、フィールドの順番です
（テンプレート等によって使用される）。
「キーの位置」には、そのファイルに対する
キー・フィールドを定義します。

ファイルを実行可能にする



LANSA は、ファイル・レベルの変更を自動的に検出
フィールド・レベルの変更には、影響分析機能を使用

ファイルのコンパイル

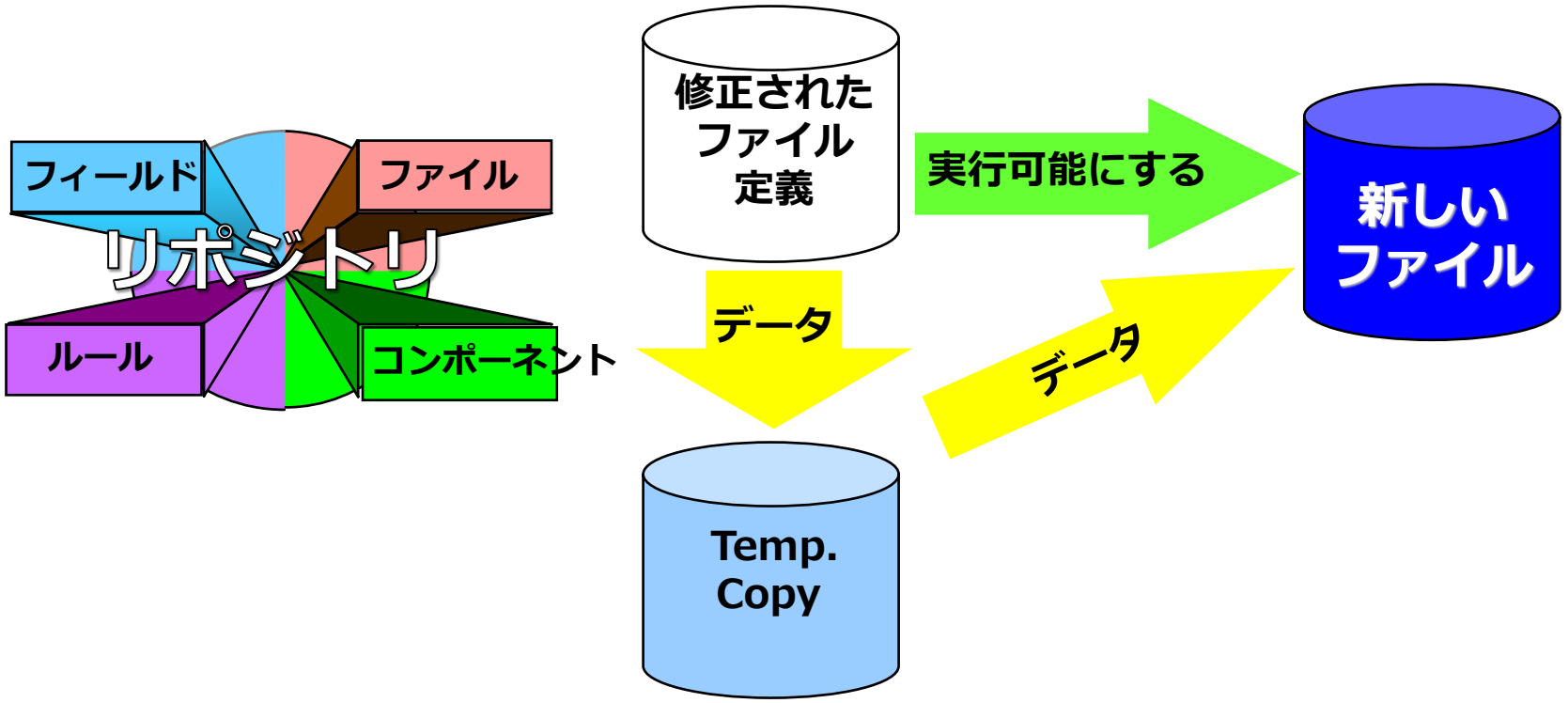


ファイルを実行可能にする：

- **テーブル／論理 ビュー／OAM の作成**

上記のうちどれを実行するかを指定することができます。あるいは、自動作成機能を使用して、加えられた変更を検出することができます。
データを保存し、再ロードすることができます。

データは？

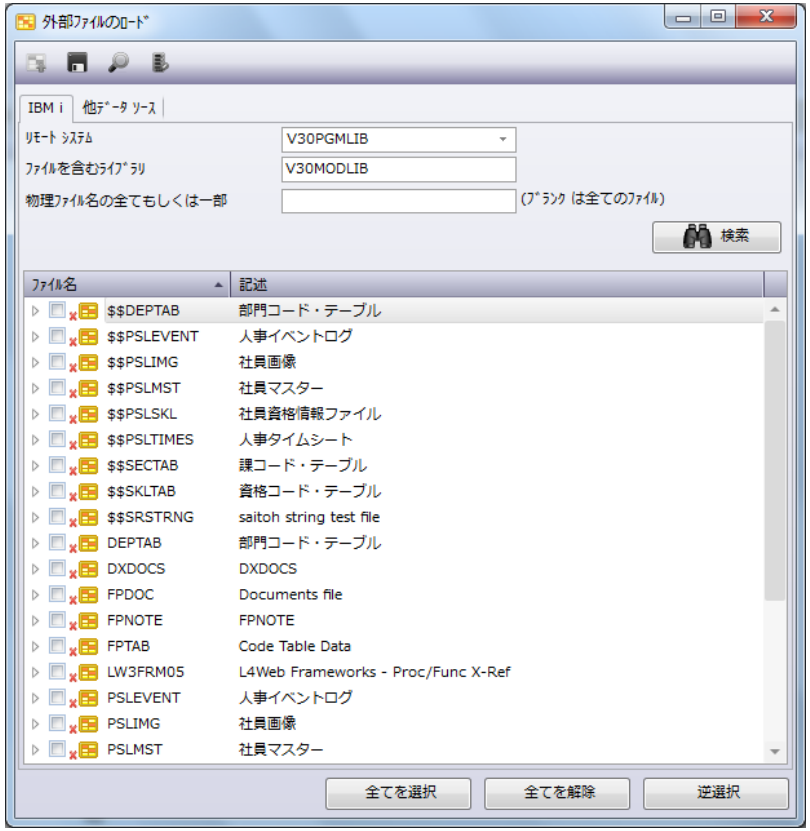


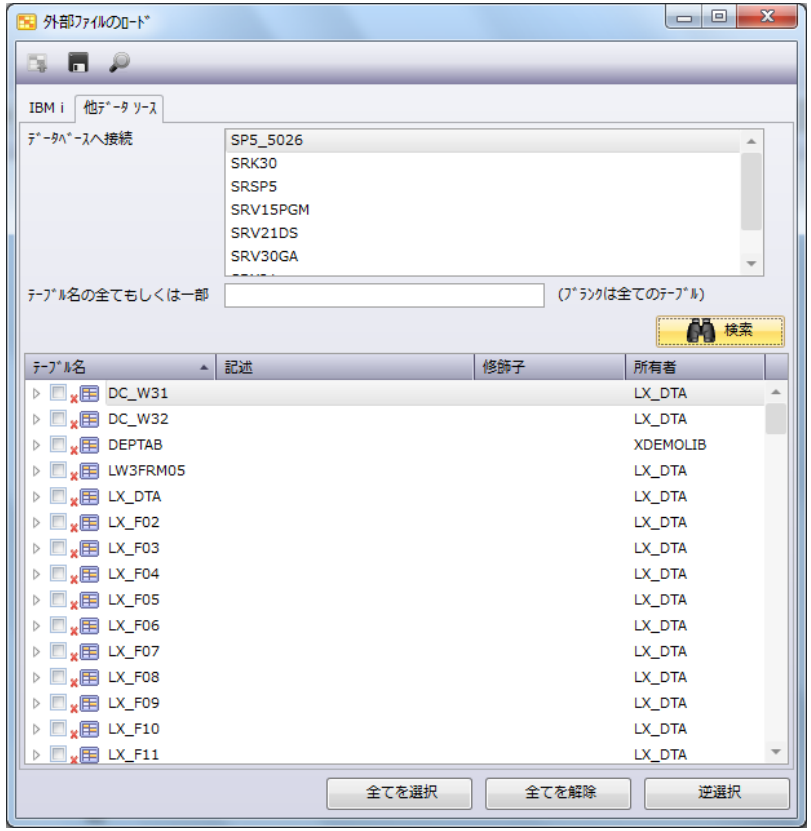
テーブル・データの保存、
および復元オプションが使用可能
LANSA は、新しいファイルにデータをコピーする

ファイル定義属性

- 特定のプラットフォーム用に、ファイル特性を最適化します。
- 複数言語定義を入力することができます。

外部ファイルのロード





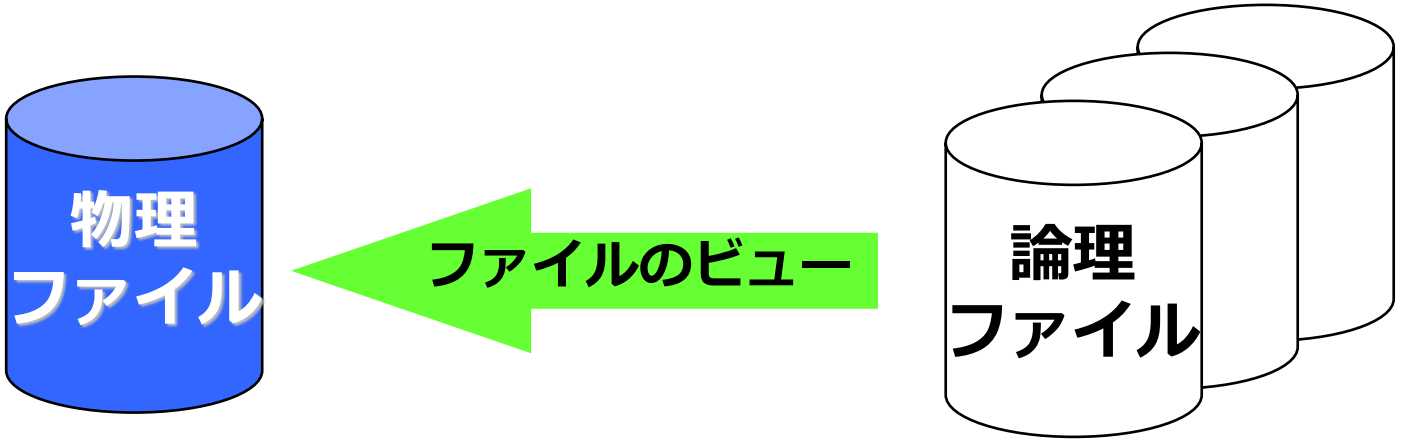
IBM i からファイルを ロードすることができます。

他データベース から ファイルをロードすることができます。

演習2:ファイル作成

c.論理ビュー

論理ビューとは？



物理ファイル

- ファイル・データを含む
- 論理ビューより先に存在して
  いなければならない

論理ビュー：

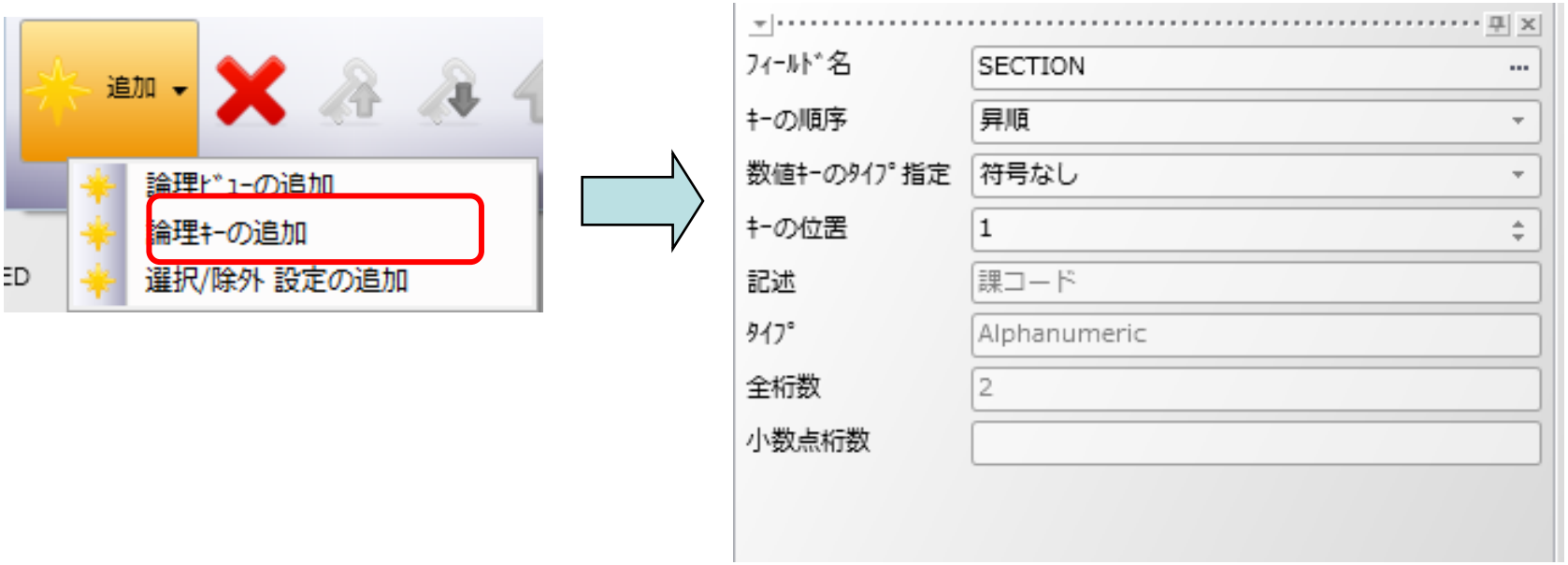
- データは含まない
- 物理ファイルの後で作成される
- 物理ファイルのビュー
- 多数のビューを持つことが可能
- 選択／除外を使用できる

論理ビューの処理

論理ビューは、
以下によって作成できます。

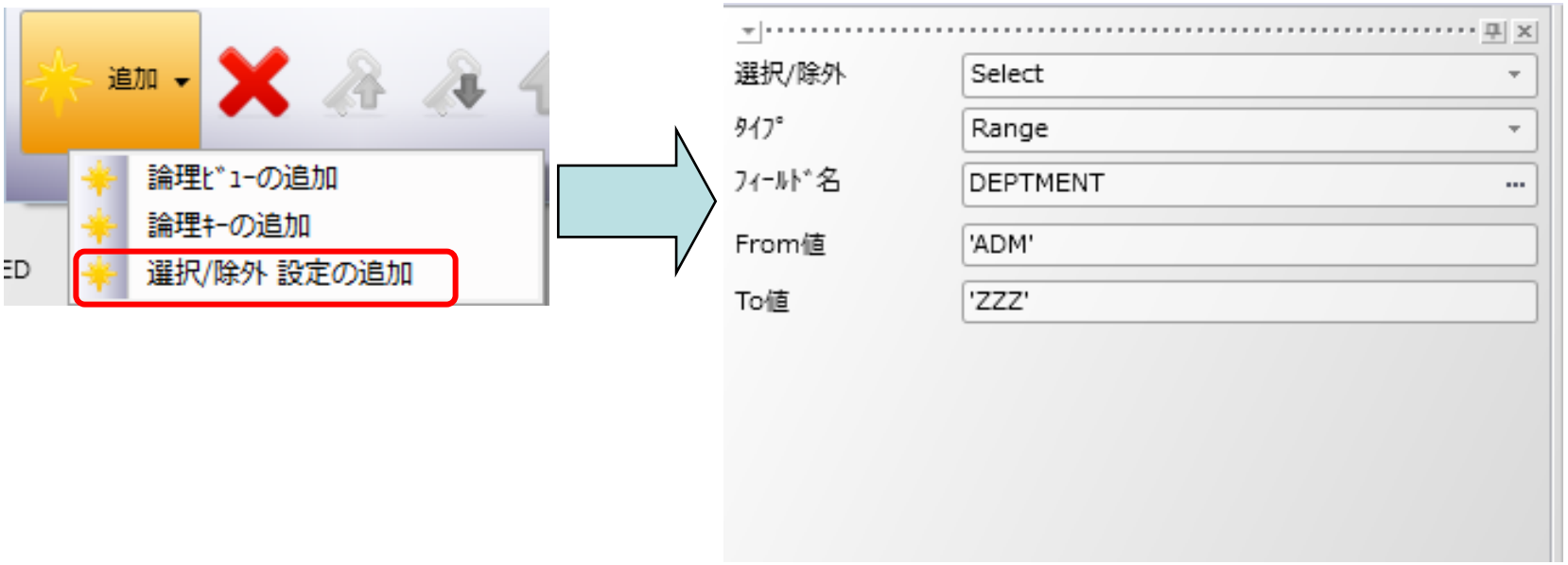
- **マニュアルで作成**
- **ファイルのコピーで作成**
- **外部ファイルのロードで作成**

論理ビューのキーフィールド



- 論理ビュー（論理ファイル）には
  キーフィールドが必要となります。
- プロンプトを使用して、
  （物理）ファイル内のフィールドの一覧から、
  キーフィールドを選択することができます。

選択 / 除外条件



選択／除外の設定

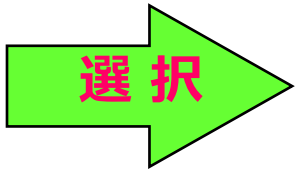
- 選択(Select)または除外(Omit)の条件を設定できます。
- オペレーションのタイプごとに、複数の条件を追加できます。

選択 / 除外条件はどのように働くのか？

LANSA

AND/OR	選択／除外	フィールド	オペレーション
	SELECT	FIELD1	COMP(EQ 'AA')

FIELD1	FIELD2	FIELD3
AA	X	10
AA	Y	20
BB	Y	30
CC	Z	10



COMP( )
RANGE( )
VALUES ( )

FIELD1	FIELD2	FIELD3
AA	X	10
AA	Y	20

頻繁に使用される場合は、DYNAMIC=NO に設定
パフォーマンスの問題に注意

論理ビューに関する考慮事項

- DBMS の制限を理解する
- DBMS のパフォーマンスに関する問題を理解する
- 論理ビューの数を制限する
- RDML には、強力なアクセス技法があります
- AS/400 上で OPNQRYF を使用して検索
- IMMED アクセス・パス （AS/400） には注意が必要

演習3:論理ビュー作成

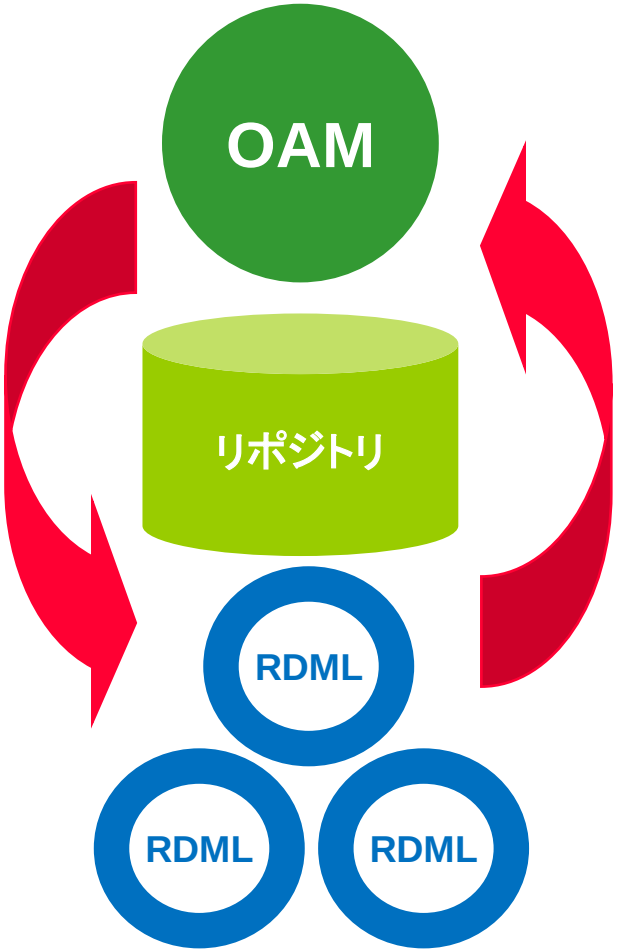
RDMLとテンプレート

詳細な内容は「LANSA RDML」コースで取り扱います。

LANSA アプリケーションを構築する

ステップ 1:
リポジトリを構築する:
フィールド、ファイル、
トリガー、ルール、
リレーションなど. . .

ステップ 2:
アプリケーション・プログラムまたはロジックを構築する。



開発は循環的なものです。RDMLの開発に応じて、リポジトリを拡張することができます。

なぜ4GL / RDMLなのか?


```
Function ...
Begin_Loop
            Change ...
            Request ...
            Insert ...
End_Loop
```

ハイレベルな
アプリケーション定義

- 高い生産性（リポジトリ＋４GL）
- 保守の負担を減らす（コード行数が減る）
- プラットフォームに依存しない定義（複数の言語に対応）
- ポータブルで耐久性がある（複数の言語に対応）
- ビジネス・アプリケーション・ロジックに集中できる
  （使いやすさ）

RDMLの構造

例：簡単な顧客ファイル・データ入力プログラム

```
FUNCTION OPTION(*DIRECT)

GROUP_BY NAME(#CUSTOMER)
                FIELDS (#CUSTNO #NAME  #ADDR1 #CITY
                        #STATE #PHONE)

BEGIN_LOOP

    REQUEST FIELDS(#CUSTOMER) DESIGN(*DOWN)

    INSERT FIELDS(#CUSTOMER) TO_FILE(CUSTMST)

    CHANGE FIELD(#CUSTOMER) TO(*DEFAULT)

END_LOOP
```

プログラミングの生産性

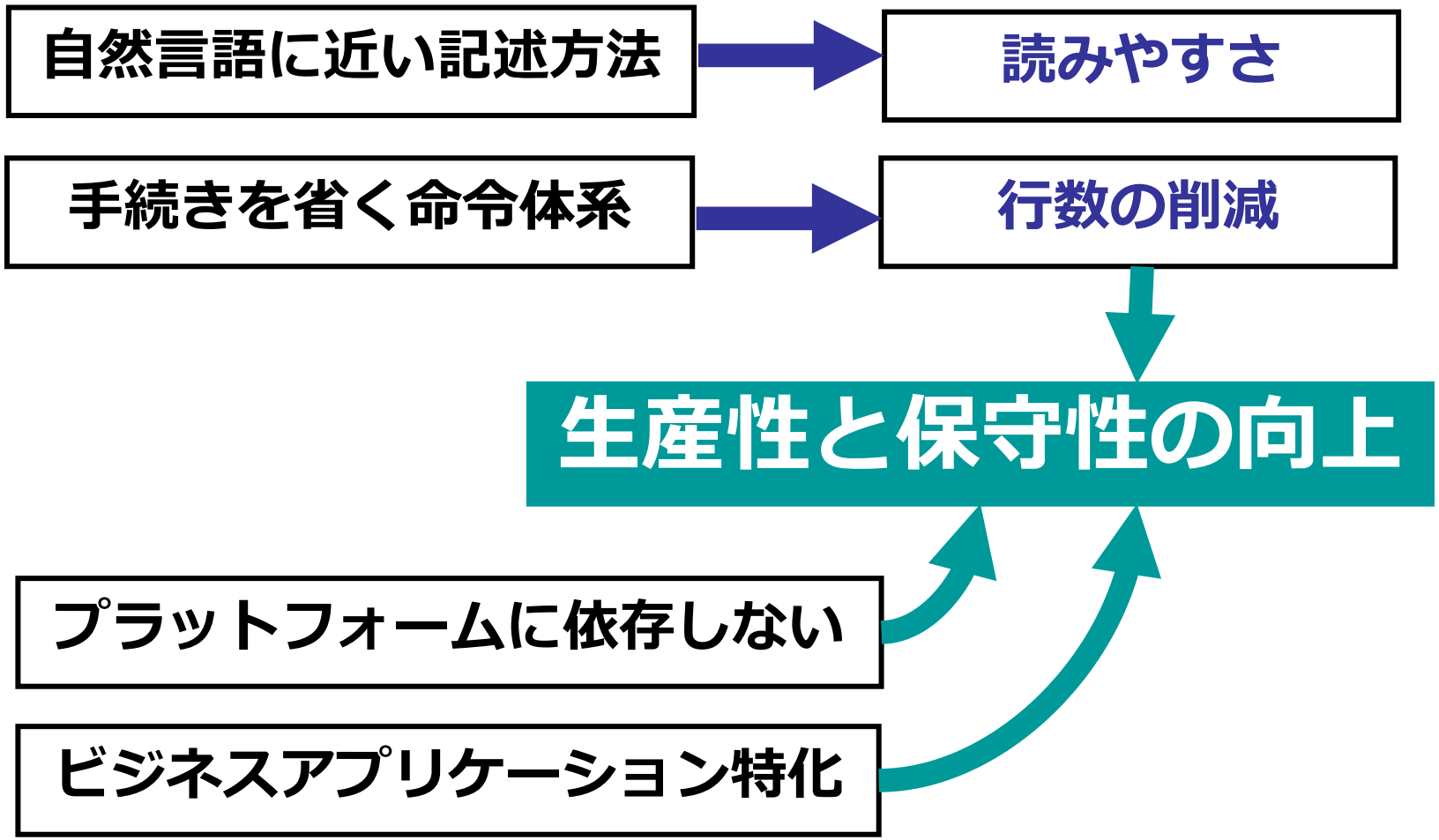
```
FUNCTION OPTION(*DIRECT)
GROUP_BY NAME(#CUSTOMER) FIELDS (#CUSTNO #NAME #ADDR1
    #CITY          #STATE #PHONE)
BEGIN_LOOP
    REQUEST FIELDS(#CUSTOMER) DESIGN(*DOWN)
    INSERT FIELDS(#CUSTOMER) TO_FILE(CUSTMST)
    CHANGE FIELD(#CUSTOMER) TO(*DEFAULT)
END_LOOP
```

開発担当者は、以下のものをコーディングする必要がないために、時間をかけることがありません：

- **ファイル宣言（オープン／クローズ）**
- **変数宣言**
- **妥当性検査ルールおよびエラー処理ルーチン**
- **ファンクション・キーおよびメニュー処理**
- **インジケータおよび画面処理制御**

等々．．．

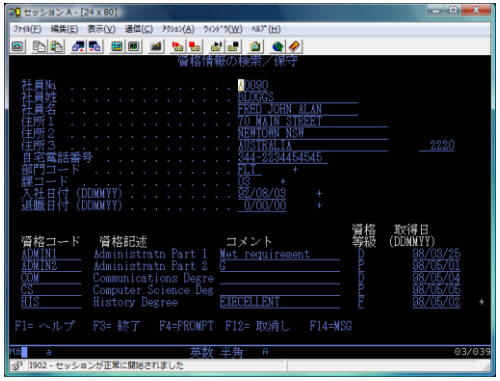
RDML のメリット


自然言語に近い記述方法
読みやすさ
手続きを省く命令体系
行数の削減
生産性と保守性の向上
プラットフォームに依存しない
ビジネスアプリケーション特化

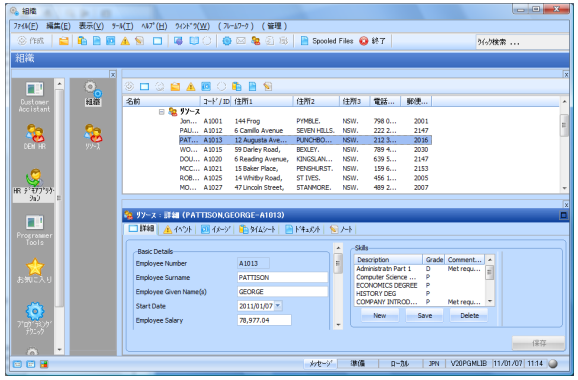
RDMLで作成できるもの

LANSA

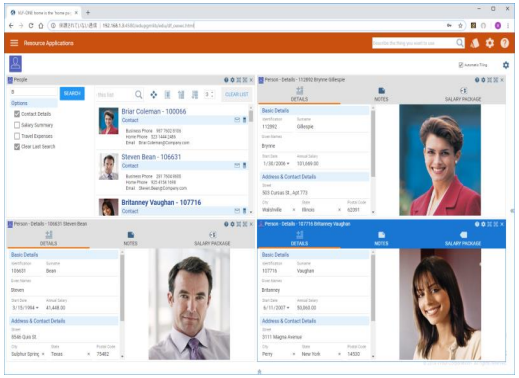
RDML



5250



GUI



WEB

RDMLは何に使用されるか?

RDMLは以下のものを構築するために使用されます：

- アプリケーション・プログラム （LANSA ファンクション）
- トリガー・ファンクション
- 呼び出される妥当性検査ファンクション
- 組込み関数
- システム変数

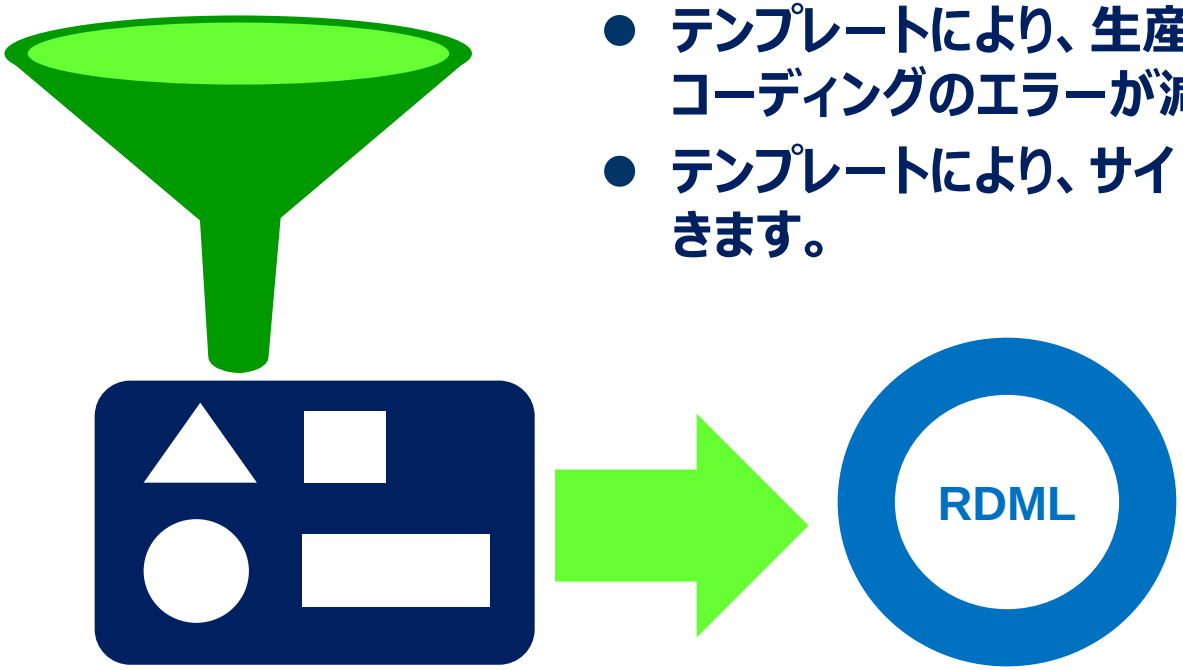
RDMLは、クライアントまたはサーバー側で実行可能です。
リポジトリの処理には、プラットフォームに依存しない開発言語が必要です。

LANSA テンプレート

どのファイルを使用しますか？

どのフィールドを使用しますか？

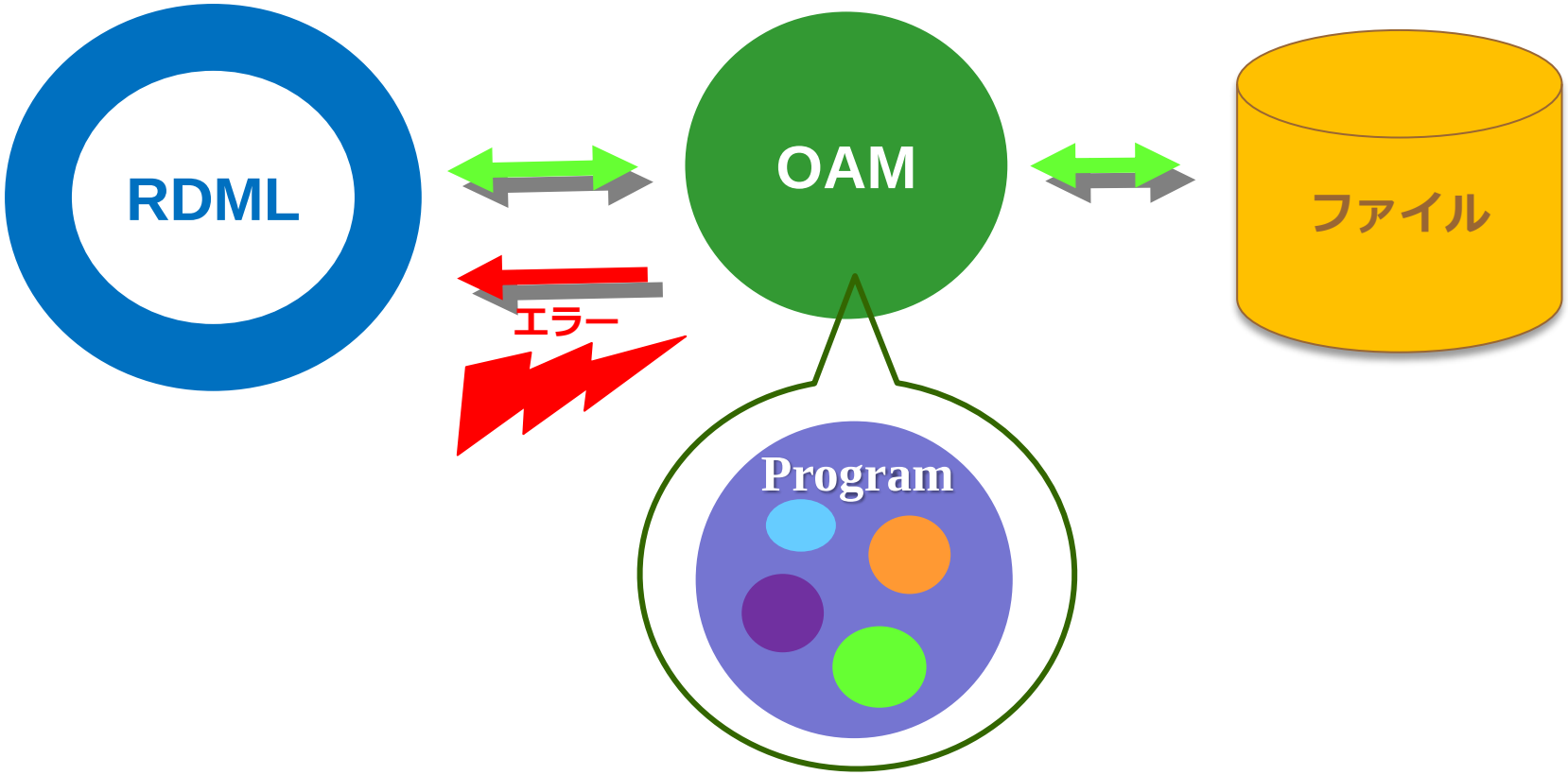
どのオプションを使用しますか？



- LANSAには、プログラミング可能なアプリケーション・テンプレートがあります。
- テンプレートは、ユーザーとの間で質問と応答をやりとりすることで、標準的なRDMLコード・ブロックを生成します。
- テンプレートにより、生産性が向上し、コーディングのエラーが減ります。
- テンプレートにより、サイト標準を守ることができます。

リポジトリとRDMLの構造

RDMLはOAMを通じてファイルI/Oを行います

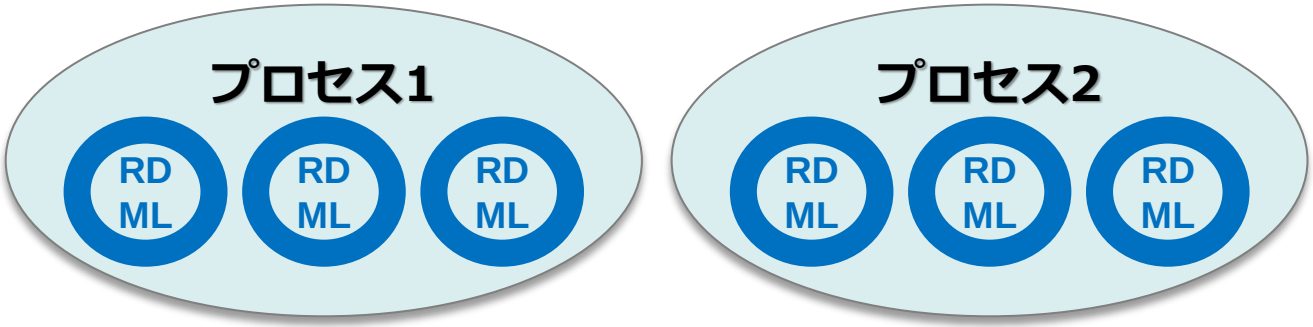


OAMにはビジネスロジックが格納されています

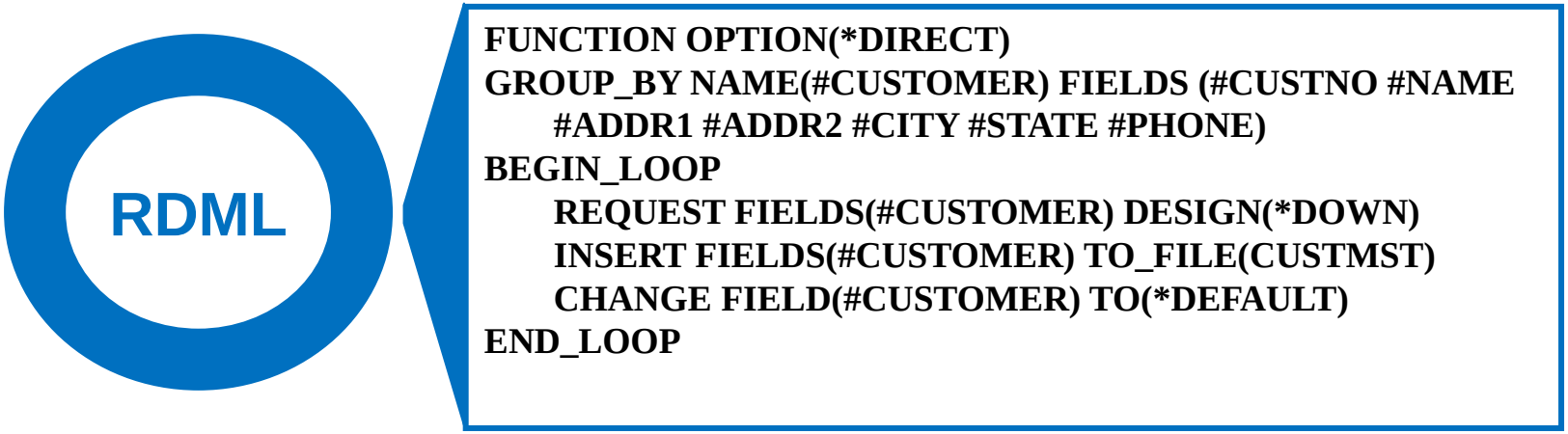
LANSA プロセス / ファンクション

プロセスとは、関連するファンクションのメニューまたはグループです。

- プロセスはファンクションを含み、ファンクションへのインターフェースを制御します。メニューまたはアクションバーとして表示されます。コードは含まれません。



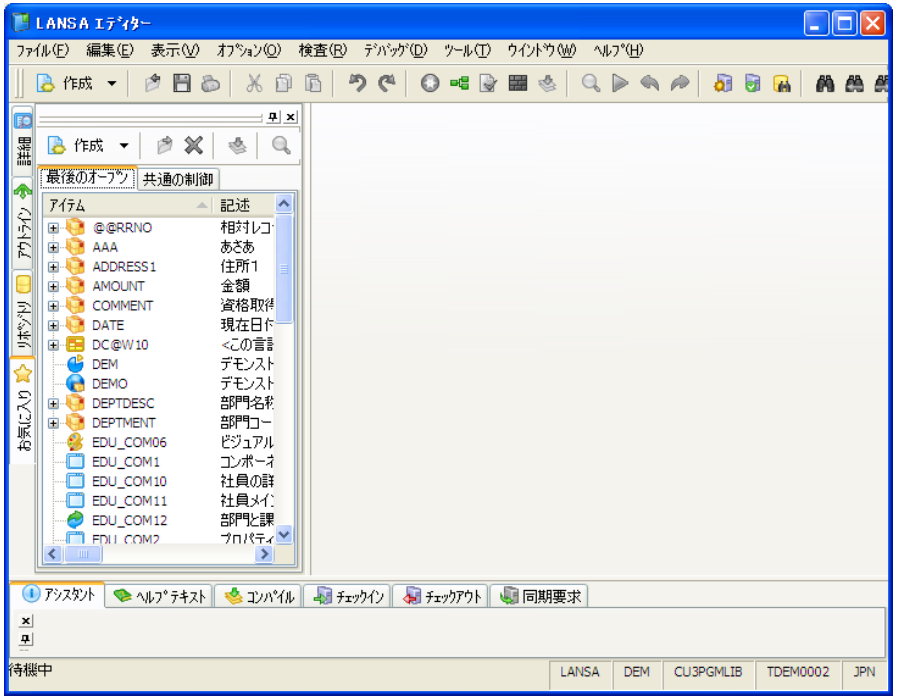
ファンクションとは、プログラムまたはRDMLコマンドのグループです。



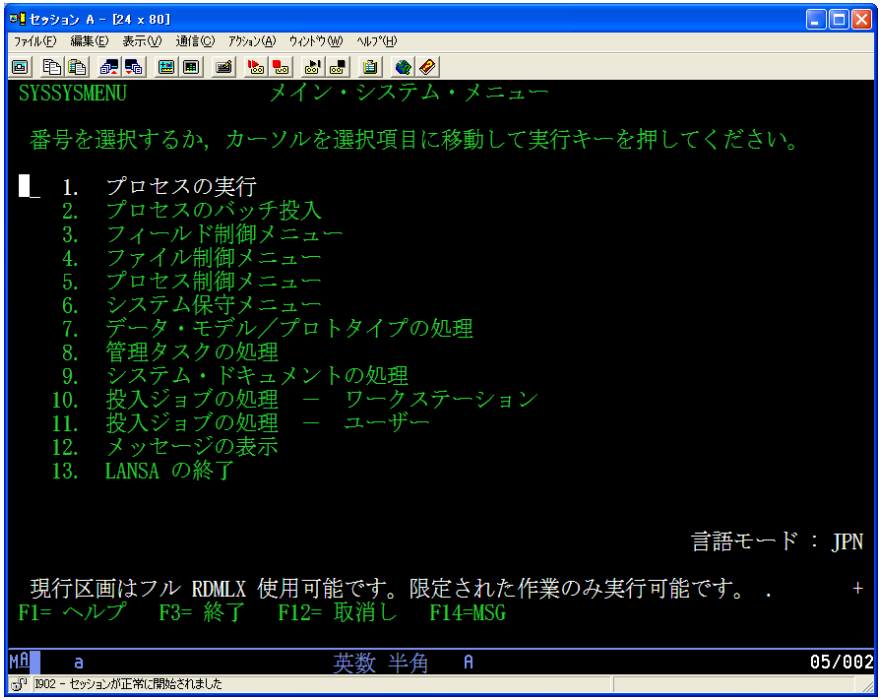
演習4：
プロセス／ファンクションの作成

開発環境の構築と運用

LANSA 開発環境の種類

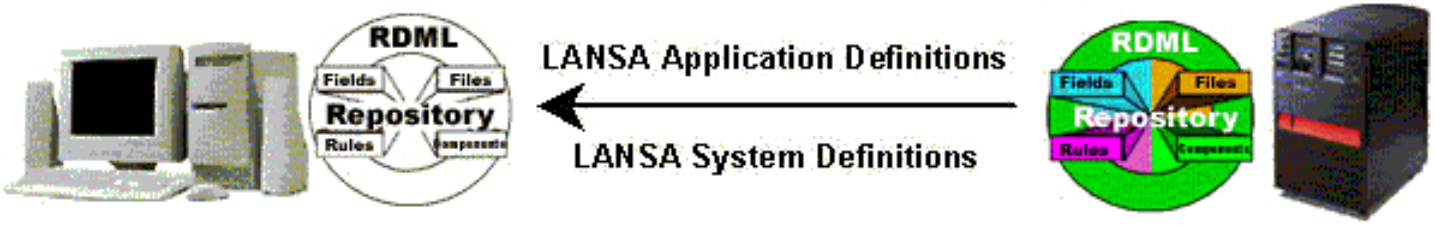


Visual LANSA



LANSA AD

マスターシステムとスレーブシステム



スレーブシステム

× 区画定義
× ＩＭＰＯＲＴ
○ 開発

マスターシステム

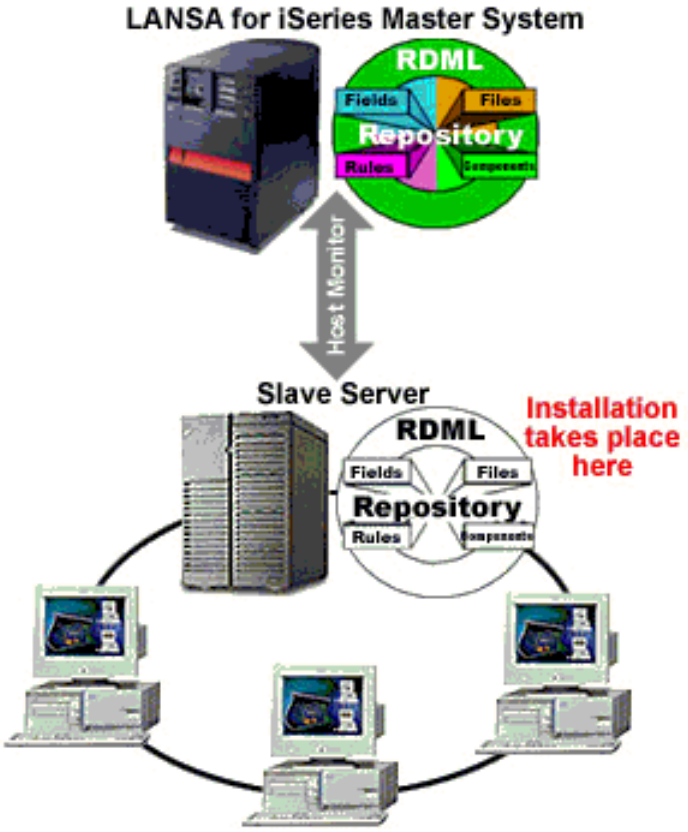
○ 区画定義
○ ＩＭＰＯＲＴ
○ 開発

スレーブワークステーション

PC へ、Visual LANSA の
開発・実行環境を
インストールします。

PC 上にリポジトリがありますが、
正のリポジトリとは認識していません。

スレーブサーバー

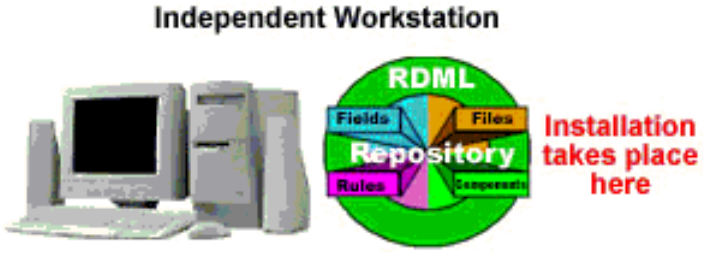


PC へ、Visual LANSA の開発・実行環境をインストールします。

PC 上にリポジトリがありますが、正のリポジトリとは認識していません。

開発もできますが、クライアントPCへのリポジトリを提供するためと意識されています。

スタンドアローンワークステーション

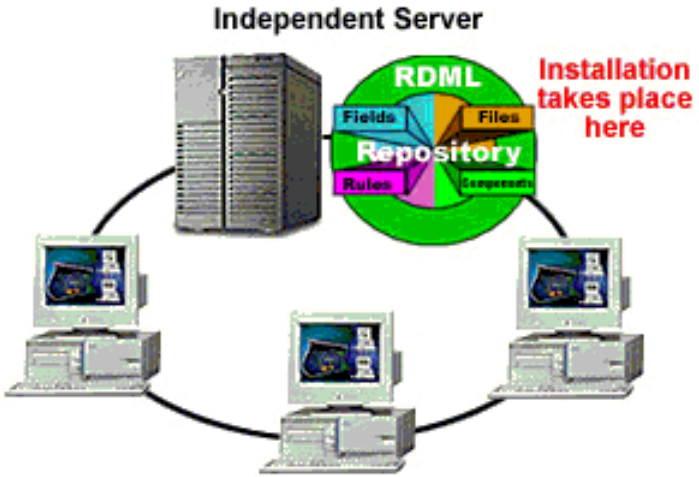


PC へ、Visual LANSA の開発・実行環境をインストールします。

PC 上にリポジトリがあり、正のリポジトリとされています。

個人で開発することを目的としており、チームで開発されることはありません。

スタンドアローンサーバー



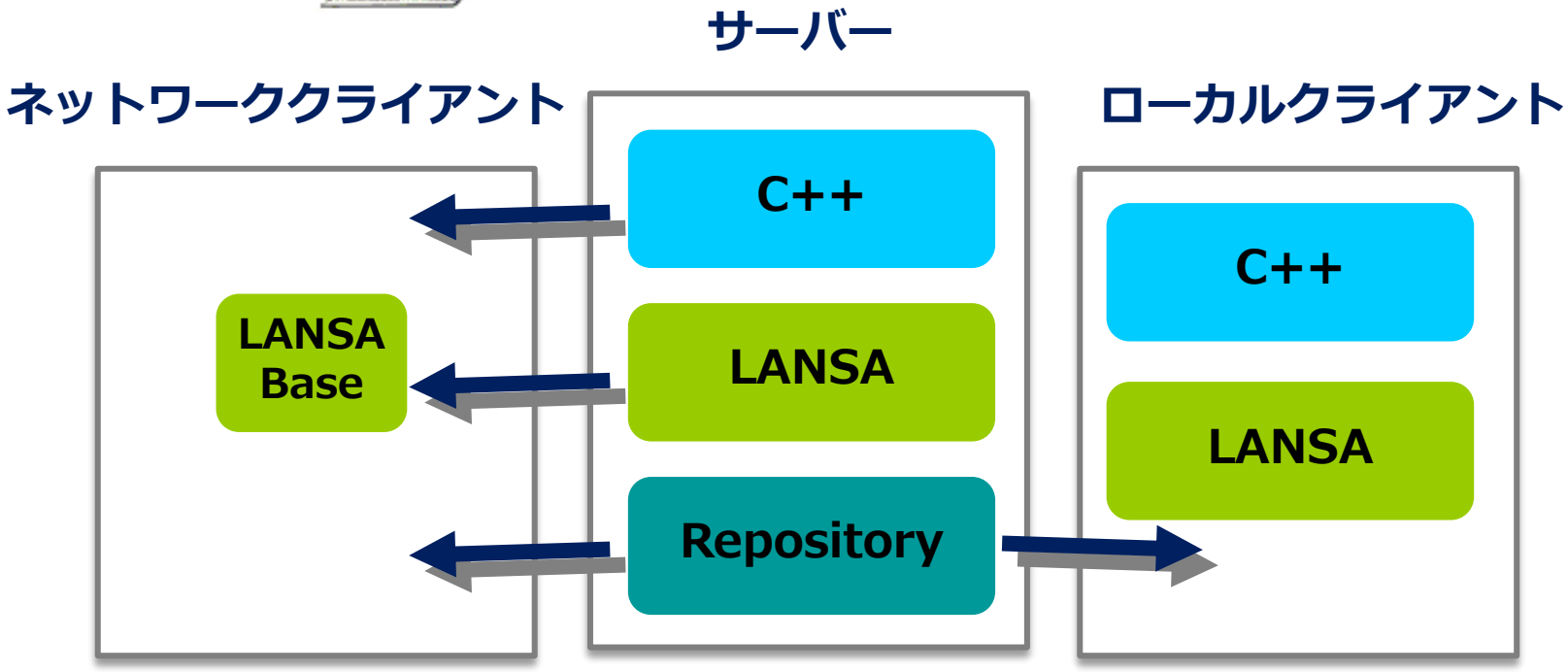
PC へ、Visual LANSA の開発・実行環境をインストールします。

PC 上にリポジトリがあり、正のリポジトリとされています。

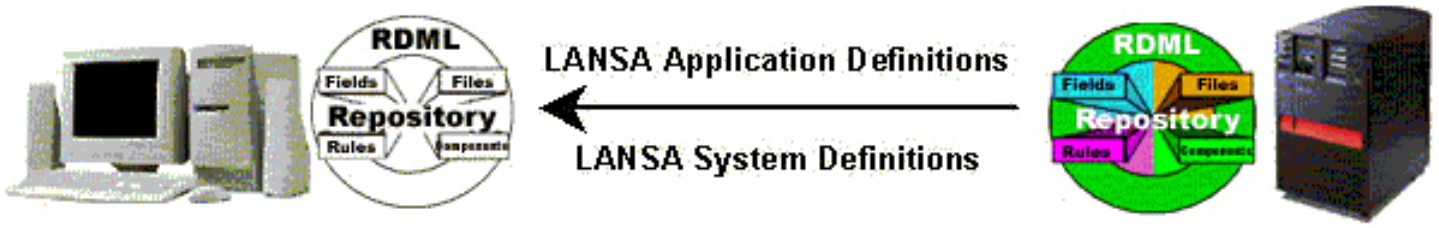
iSeries を使用しない開発環境です。

クライアント


Visual LANSA System
RDML
Fields
Files
Repository
Rules
Components
RDML
Fields
Files
Repository
Rules
Components
Installation takes place here


サーバー
ネットワーククライアント
ローカルクライアント
C++
LANSA Base
LANSA
Repository
C++
LANSA

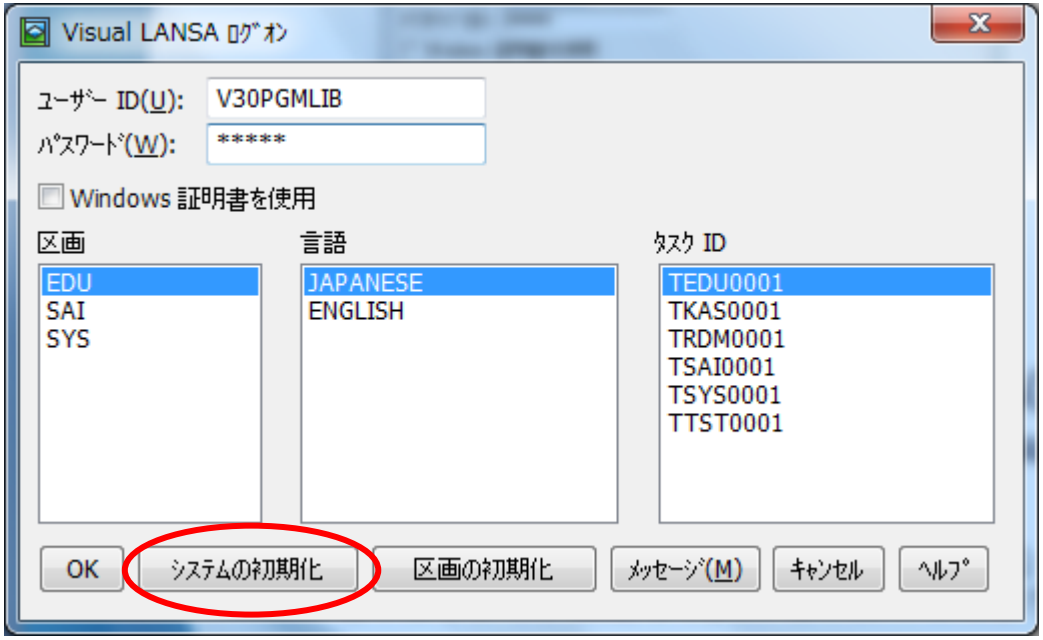
開発環境の同期



スレーブシステムの開発環境の定義を、マスターシステムの開発環境の定義と同期する必要があります。

スレーブシステムは、単独で環境情報を定義することができません。区画名、ライブラリー、コンパイルオプション、フィールドタイプなど、開発に関わる情報は、マスターシステムで定義し、スレーブシステムに同期する必要があります。

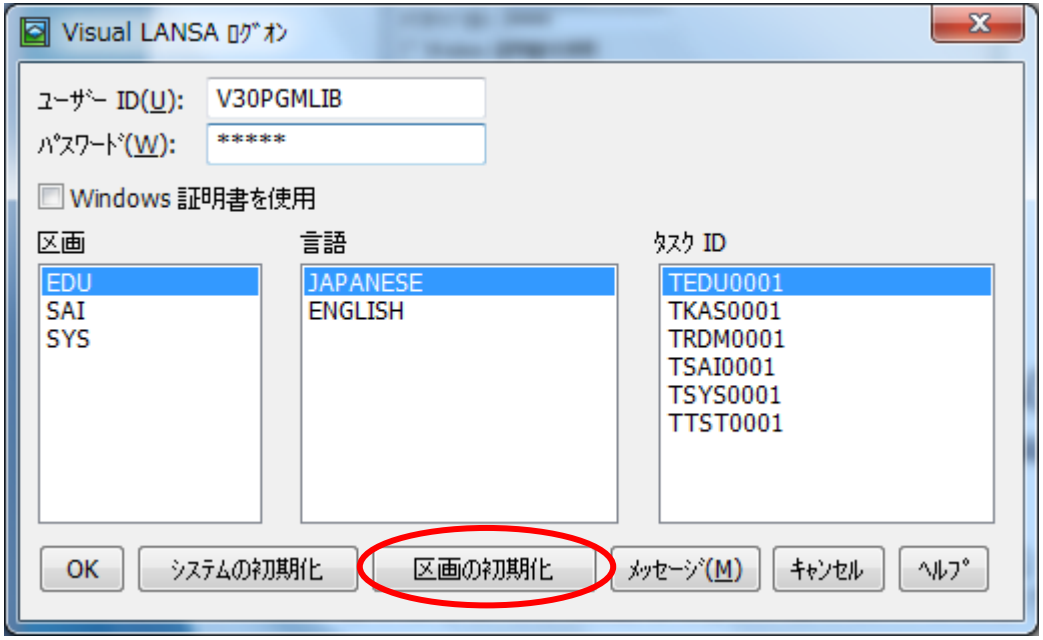
システムの初期化



システムの初期化は、マスターシステム(LANSA AD)のシステム定義をスレーブシステム(Visual LANSA)に更新するために使用されます。

- **すべてのLANSAシステム情報(区画、言語、ユーザー、タスク、セキュリティ、設定など)は、LANSA ADにより保守されます**

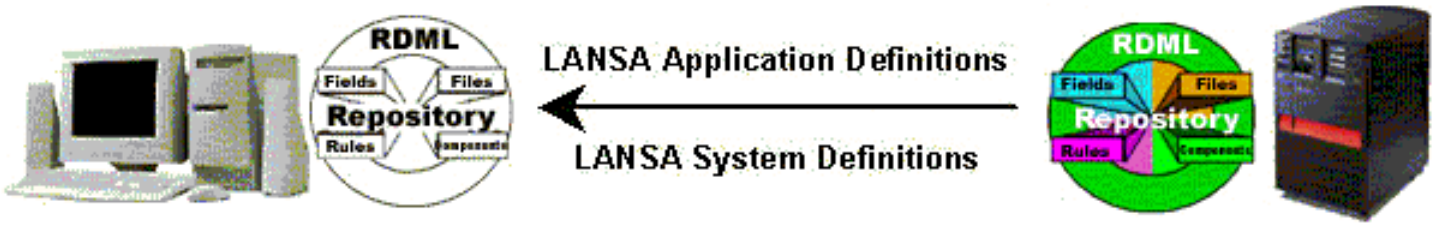
区画の初期化



区画の初期化は、Visual LANSA上の個々の区画で使用可能な機能をインポートするために使用されます。

- 使用可能な機能は、区画ごとに設定されます。
- 新規に区画を作成した場合や、機能を有効にしたい場合は、個々の区画ごとに初期化を行う必要があります。

リポジトリの同期



スレーブシステムを使用して開発を行う場合、スレーブシステムとマスターシステムのリポジトリを同期する必要があります。

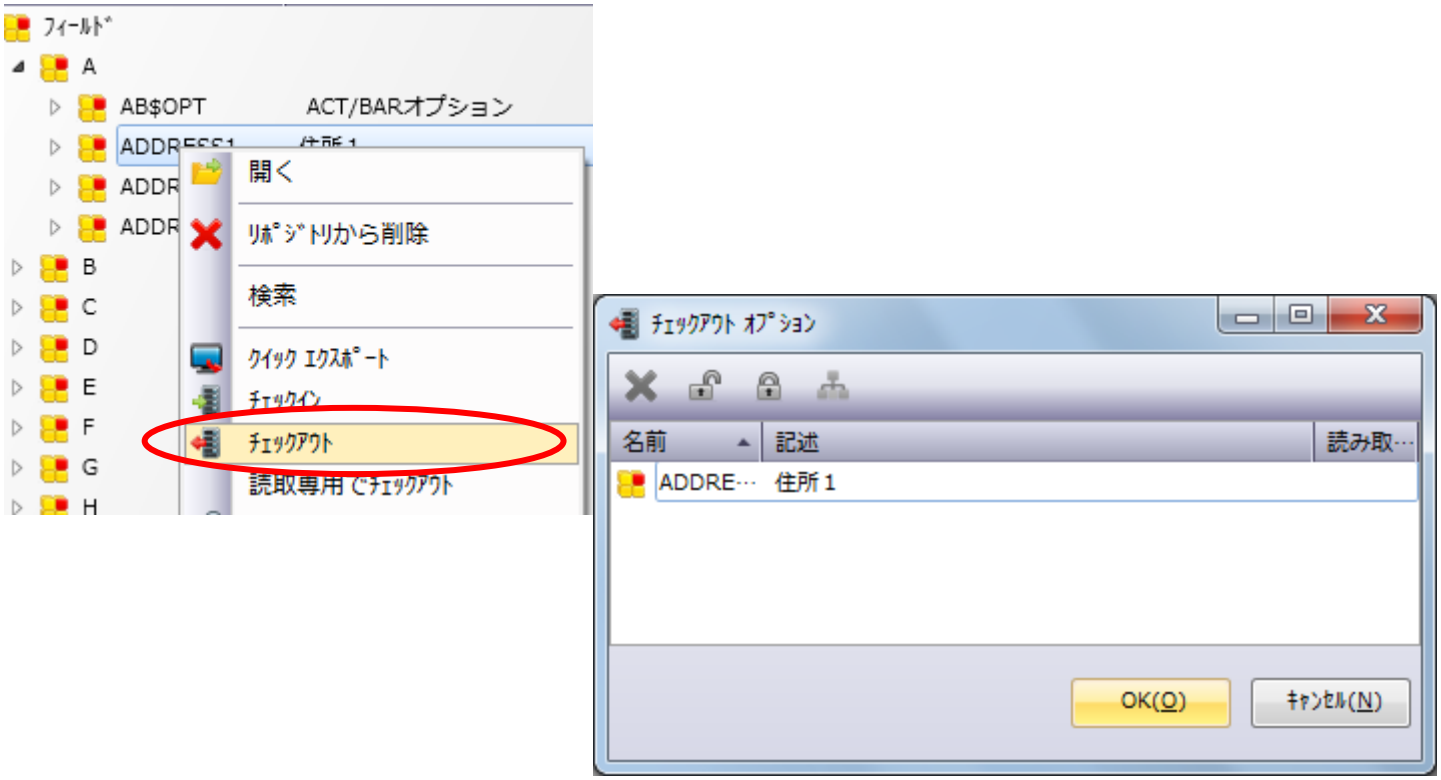
例えば、スレーブシステムでフィールドを新規作成したら、そのフィールド定義をマスターシステムへ送らなければなりません。その逆もしかりです。

ファイルやプロセス／ファンクションも同様に、一方のシステムで作成した定義を、もう一方のシステムへ送る必要があります。

送るのはあくまでも定義です。オブジェクトは送ることができません。

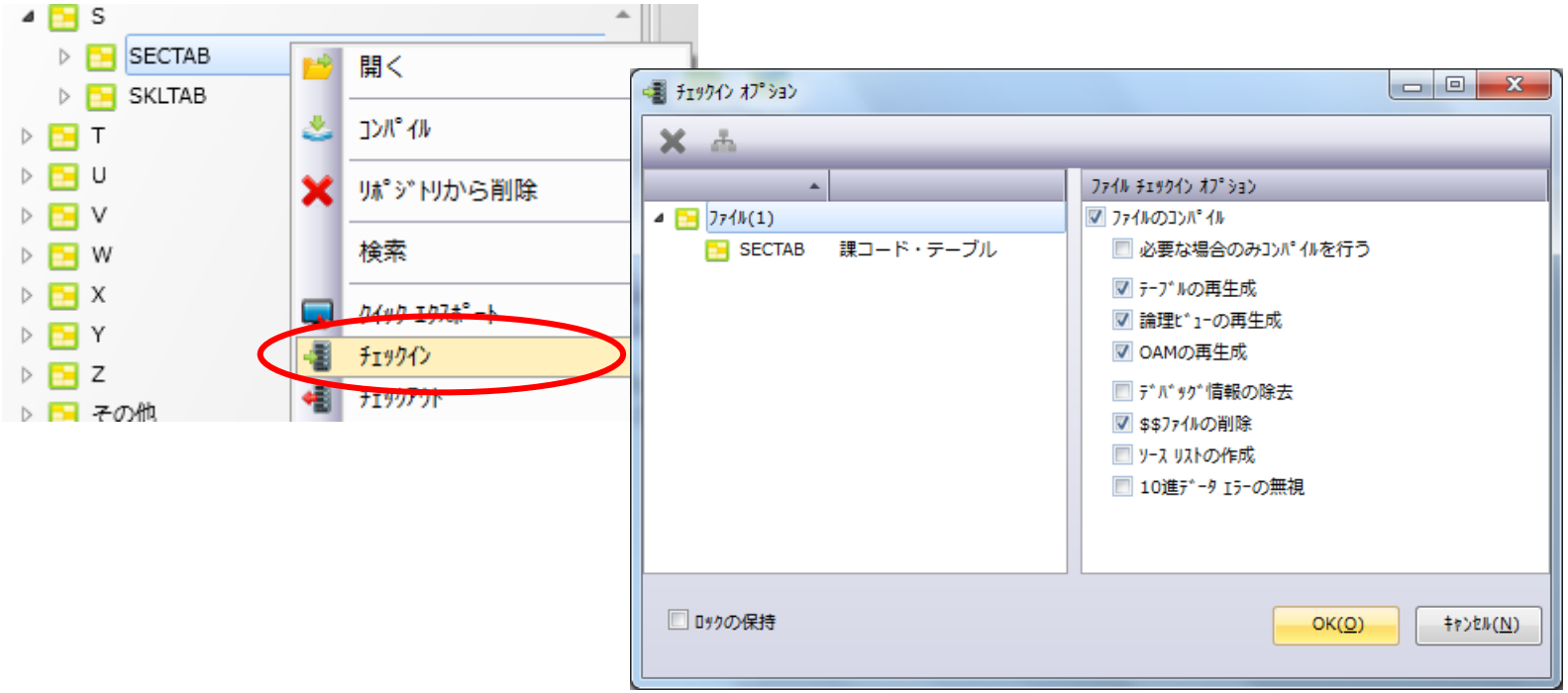
オブジェクトは各々のシステム上でコンパイルされ、作成されます。

チェックアウト



LANSA AD システムにあるオブジェクト定義をVisual LANSA にコピーする機能をチェックアウトと呼びます

チェックイン



Visual LANSA で作成・変更したオブジェクト定義を LANSA AD のマスタリポジトリに置く行為をチェックインと呼びます。

チェックインでは、マスターリポジトリのプラットフォームでのコンパイルを、指示することができます。

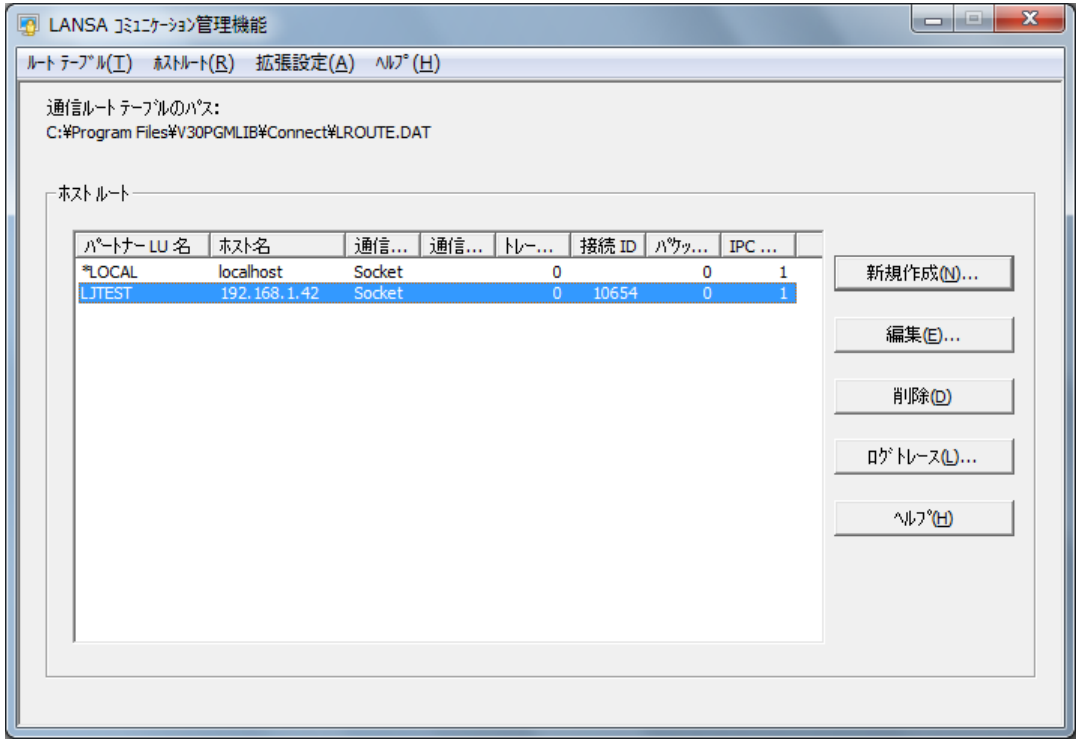
ホストモニター

ホストモニターは、マスターシステム(LANSA AD) とスレーブシステム(Visual LANSA) をつなぐためのものです。以下の事を念頭においてください。

- LANSA/ADは、すべての開発オブジェクト(アプリケーション・フィールド、ファイル、フォームなど)を含むマスター・リポジトリをホストする
- 複数のVisual LANSA開発環境(スレーブ・システム)がLANSA/ADマスター・システムに接続できる

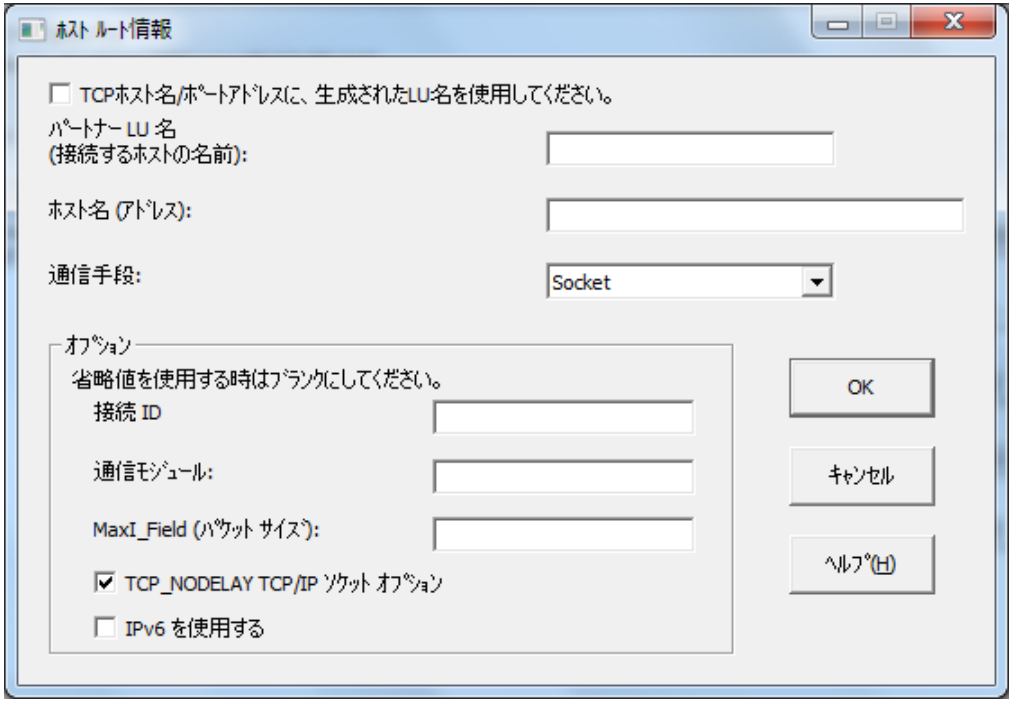
LANSAコミュニケーション管理機能

- TCP/IPを使用してSystem iまたはWindowsサーバーに接続する場合、LANSAコミュニケーション管理機能で通信先を登録する必要があります。
  - 前提：System i及びPCでTCP/IPが正しく構成されている。“PING”コマンド等で接続を確認



ホストルート情報

- 接続するサーバーへのルート情報を登録します。
- 指定する項目は以下の通りです。
  - パートナーLU名
  - ホスト名
  - 接続ID(省略値4545)



リスナー・ジョブ

- System i上で、クライアントからの接続要求を待機するジョブです。
- “LISTENER”というジョブが、各クライアントに対して、“TP000000nn”というジョブを機動させて割り当てます。
- リスナー・ジョブが起動していない場合、クライアントからの接続がエラーとなります。
- 開発と実行の両方で使用されます。

リスナー・ジョブの開始コマンド

STRSBS SBS(pgmlib/pgmlibと同名のサブシステム記述)

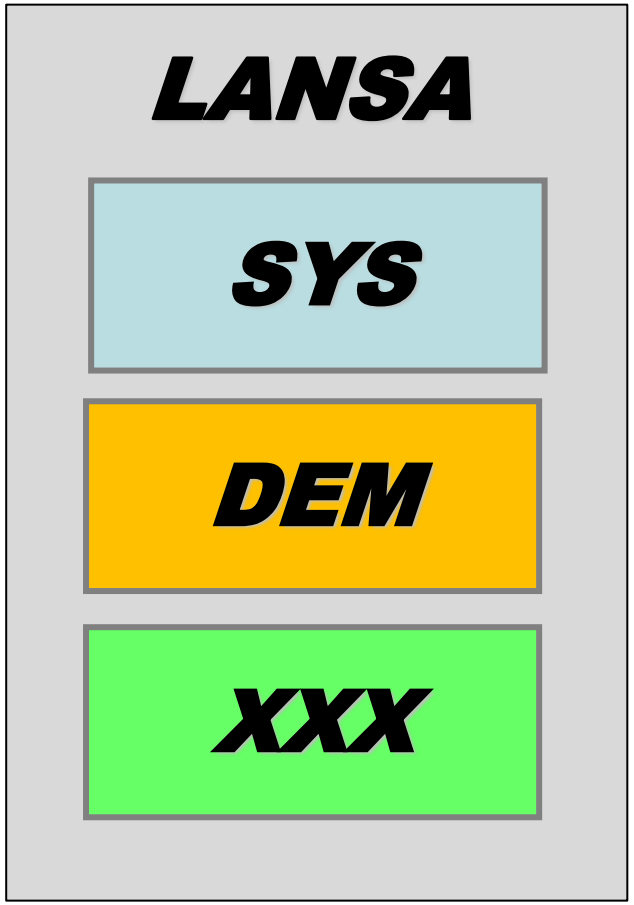
・pgmlibの省略値は”DCXPGMLIB”です。

・pgmlibと同名のサブシステム記述がインストール時に作成されます

演習5：チェックイン
演習6：チェックアウト

区画

区画とは



- LANSAの中に作成される環境の範囲です。
- 個々の区画は、他の区画から完全に独立しています。
  - 一部（システム変数等）のオブジェクトは共通オブジェクトとして区画で共有されます。
- 開発・実行は　区画毎に行われます。

区画の定義

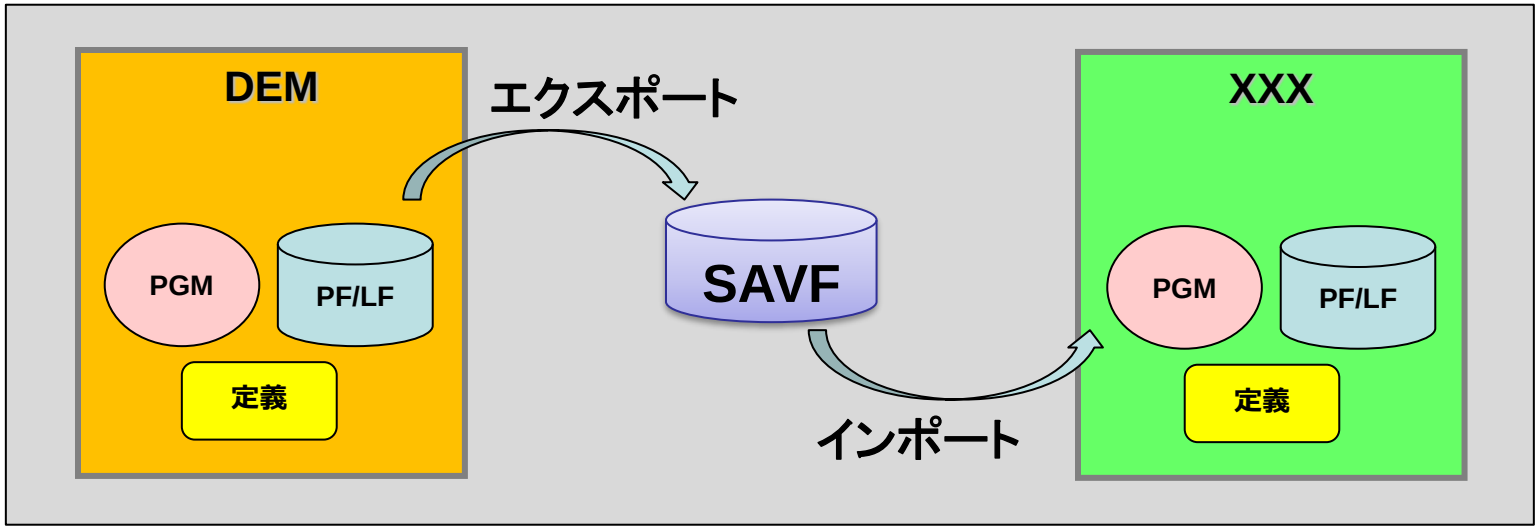
- 区画名/記述
- 機密保護担当者
  - 区画内の最高権限者を指定します。
- ライブラリー
  - 区画内のファイルとプログラムの生成ライブラリーを指定します。
- 複数言語設定
  - 区画内で使用可能な言語を指定します。
- RDMLX使用可能
  - RDMLかRDMLXの２つの区画タイプを選択します。
- フィールドタイプ/省略値
  - 使用可能なフィールドタイプと各フィールドの省略値を指定します。
- 画面の共通表示項目/ファンクションキー記述
  - UI上のスクリーンタイトルや日付などの共通表示項目の出力を設定します。
  - アプリケーションで使用するファンクションキーの割り当てと記述を設定します。
- オブジェクト権限
  - 区画内のオブジェクト定義の使用権限/データ権限を設定します。
- メニュー使用権限
  - 開発メニューの表示/非表示、使用権限を設定します。
- etc.…

区画についての注意事項

- 区画間のコミュニケーションはサポートされていません。
  - ✓ 1つの区画で定義されたフィールド、ファイル、コンポーネント、プロセス、およびファンクションは、他の区画から自動的にはアクセスできません。オブジェクトは区画間で配布またはエクスポート/インポートする必要があります。
- 各区画は個別の論理リポジトリを持ちます。
  - ✓ 複数の開発区画を持つことは、1つの開発リポジトリの共有という基本的な概念に反するので推奨されておらず、再利用が複雑になります。さらに、不必要な保守と配布手段が作成されます。
- 各区画は個別のセキュリティ・システムを持ちます。
  - ✓ 開発者は区画にログオンします。エンド・ユーザーは、区画内でアプリケーションを実行します。区画が多すぎたり、複数の区画にアクセスするユーザーの数が多すぎると、保守の問題が発生します。
- 各区画は区画内で使用できるオブジェクト、言語、生成される情報、そのほかの詳細を決定する独自の開発特性のセットを持ちます。
- 区画には、固有の1文字の識別子が必要なので、1つのLANSAシステムに作成される区画の数には制限があります。

エクスポート/インポート

区画間でのリポジトリ定義とオブジェクトの移行は、
エクスポート/インポートによって行います。



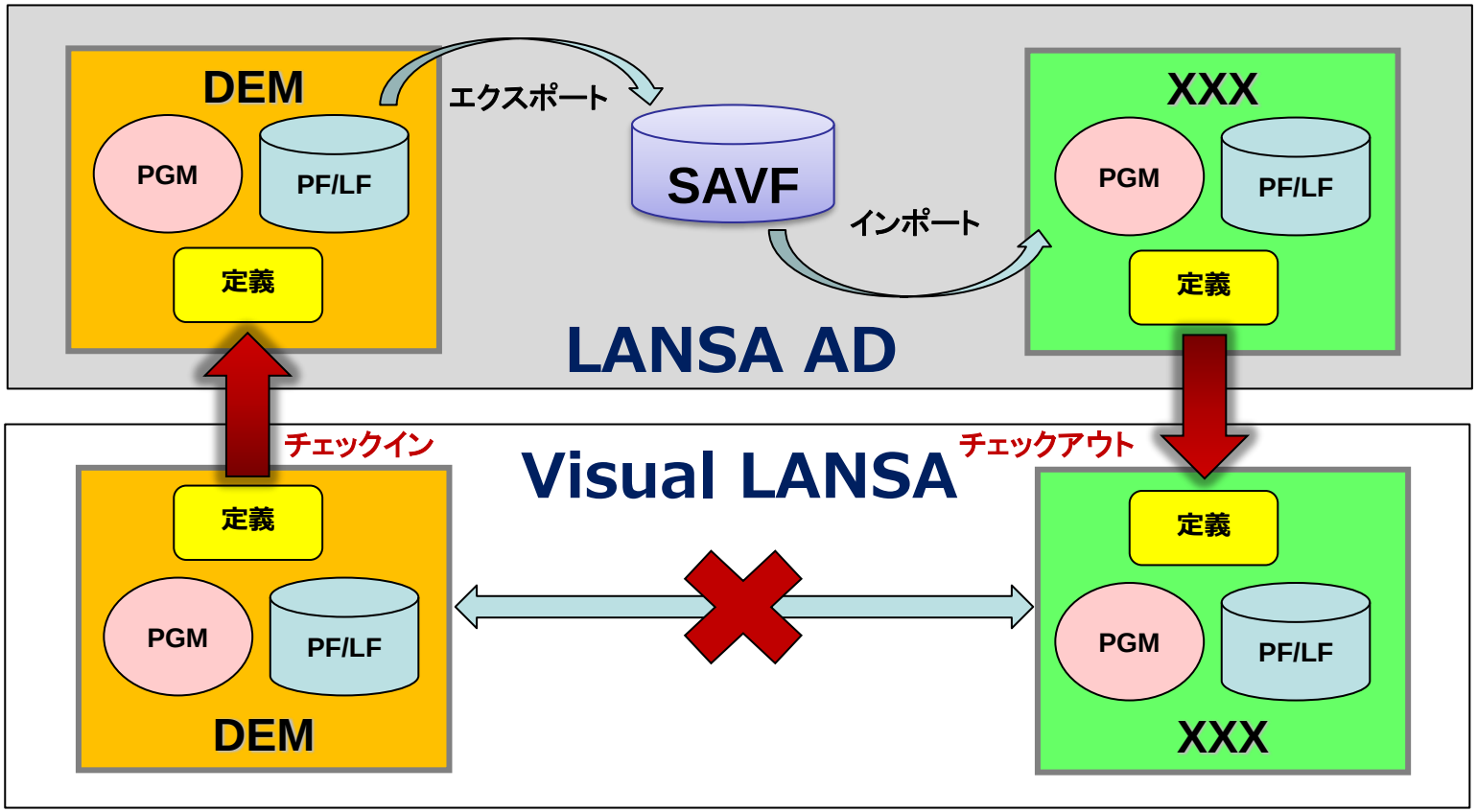
エクスポート/インポートは、LANSA ADの機能です。

Visual LANSAでは配布ツールを使用します。

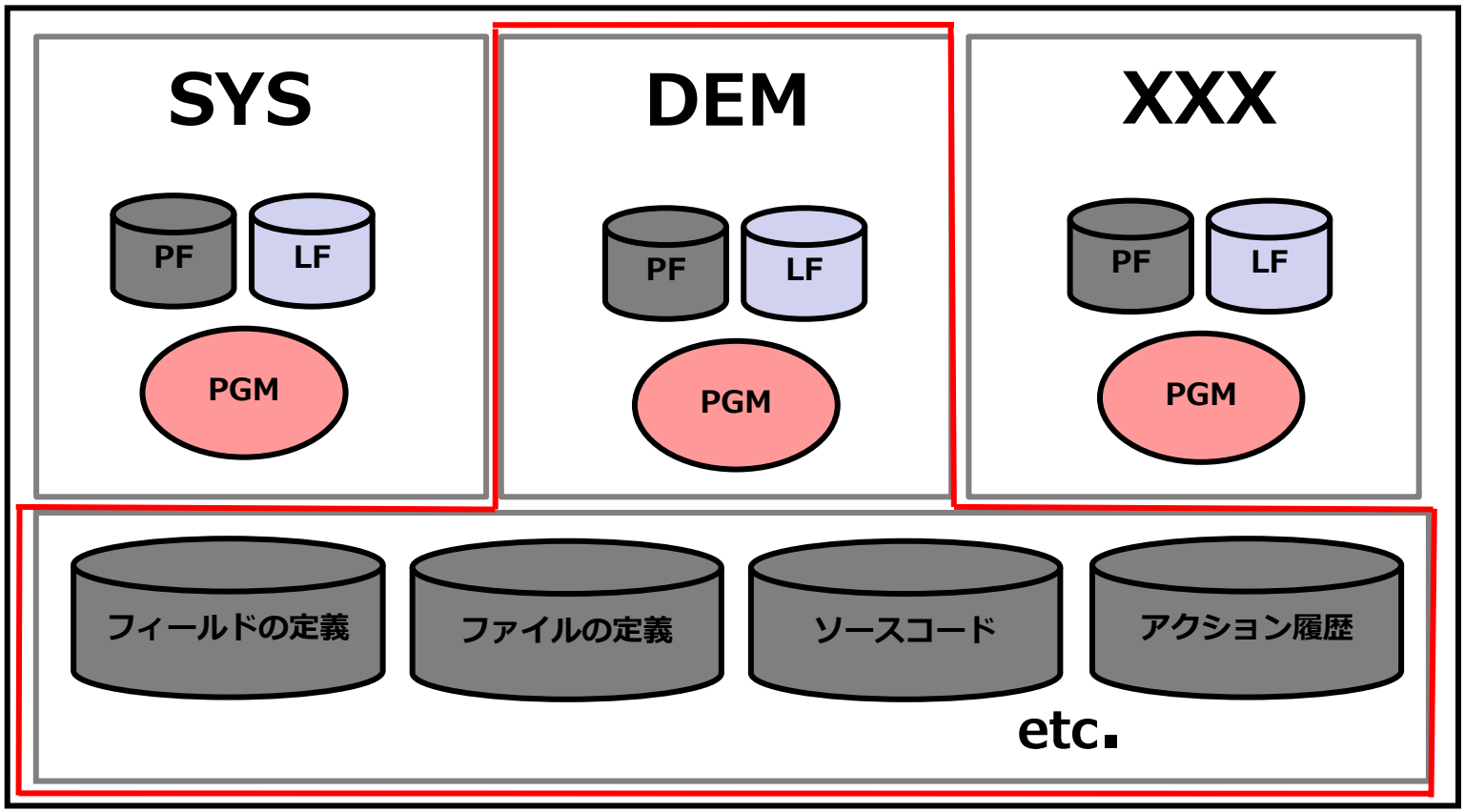
スレーブシステム上での区画間移行はサポートされていません。

スレーブシステムでの区画間移行

スレーブシステム上での区画間移行をするためには、LANSA ADにチェックインし、エクスポート/インポートで移行した後に、移行先の区画からチェックアウトを行ってください。



定義とオブジェクト



- 定義は共通のライブラリー内のデータベース・ファイルに保管されます。
- オブジェクトは区画のライブラリー（またはディレクトリ）に保管されます。

RDMLとRDMLX

詳細な内容は「LANSA RDMLX」コースで取り扱います。

違いは４つあります。

LANSA

- RDML/RDMLXオブジェクト
- RDML/RDMLXコマンド
- RDML/RDMLXシンタックス
- RDML/RDMLX区画

RDML/RDMLXオブジェクト

LANSA

- **RDMLオブジェクト**
  - IBM i上で、RPG(Ⅳ+ILE)にコンパイルされるオブジェクトです。
  - 256バイト以内の固定長文字列、10進数数値のみ使用可能です。
  - LANSA ADとVisual LANSAの両方で開発可能です。
  - 80x24の5250画面で表示することができます。
  - **RDMLXオブジェクトとの互換性がありません。**

- **RDMLXオブジェクト**
  - IBM i上で、C++にコンパイルされることを前提としたオブジェクトです。
  - SQL型のフィールドが使用可能です。
  - フィールドがオブジェクト認識されます。
  - **RDMLX区画のみで開発が可能です。**
  - RDMLXシンタックスが使用できます。
  - **80x24の5250画面で表示することができません。**
  - RDMLオブジェクトも使用することができます。

RDML/RDMLXコマンド

- **RDMLコマンド**
  - ビジネスアプリケーションに最適化された、LANSAの基本的なコマンドです。
  - データベースI/O、画面/帳票印刷、条件判断、エラーハンドルなどを処理します。
  - LANSA ADとVisual LANSAの両方でコーディング可能です。

- **RDMLXコマンド**
  - Full GUIを作成するために追加されたコマンドです。
  - GUIの画面制御とオブジェクトを処理します。
  - Visual LANSAでのみコーディング可能です。

RDMLXコマンドはRDMLオブジェクトでも使用することができます。

RDML/RDMLXシンタックス

LANSA

- **RDMLシンタックス**
  - RDMLコマンドに則したシンタックスです。

例) LB1 : SELECT　Fields (#ABCDE) From_File(SAMPPF)
　　ラベル　コマンド　　パラメータ　パラメータ値　...

- **RDMLXシンタックス**
  - ASSIGNコマンド(代入式)が使用できます。
  - 文字列の連結ができます。
  - 引用符が不要な箇所があります。
  - フィールドのプロパティ、イベント、メソッドが使用できます。
  - BOOL値の検査ができます。
  - **RDMLXオブジェクトのみで使用可能です。**

RDML/RDMLX区画

LANSA

- **RDML区画**
  - RDMLオブジェクトのみ使用可能な区画です。
  - LANSA ADとVisual LANSAの両方で開発可能な区画です。
  - GUIも開発できますが、使用できる機能が制限されます。
  - WAMとVLF(RAMP)は開発/実行することができません。
  - RDMLX区画に変更することも可能です。

- **RDMLX区画**
  - RDMLとRDMLXの両方のオブジェクトが使用可能な区画です。
  - **Visual LANSAのみで開発可能な区画です。**
    ※ LANSA ADで実行することはできます。
  - Full GUIの開発に適した区画です。
  - WAMとVLF(RAMP)を開発/実行することができます。
  - **RDML区画へ変更、または戻すことができません。**

まとめ

開発を始める前に…

- Advanced Software Made Simpleというコンセプトを
  忘れないでください。LANSAは”Simple is Best”を実現します。

- 開発の第1ステップはリポジトリの構築です。
  リポジトリは全てのアプリケーションで共有されます。

- 開発の第2ステップはRDMLの作成です。
  アプリケーションのタイプは一つではありません。業務に最適化された
  アプリケーションのタイプを知っておく必要があります。

- プロジェクトメンバーの構成や、
  アプリケーションのタイプを考慮して開発環境を構築してください。

- 開発に必要となる、より詳細なLANSAの知識を、
  専門の各コースで習得してください。