ODIP v4.0 リリースノート

2019/08/23

（株）インテリジェント・モデル

目　次

A. 変更内容

1. トランスフォーマ・リポジトリの差分リリース機能の追加

(1) 改定内容

① 概要

ODIP トランスフォーマのトランスフォーマ・リポジトリを比較するコマンド repcomp と、トランスフォーマ・リポジトリをインポートするコマンド repimp に‘-patch’オプションが追加され、2 つのトランスフォーマ・リポジトリの差分データの作成、反映ができるようになりました。

② 改定されたコマンド

表 1 のコマンドが改定されました。

表 1 差分リリース機能で改定されたコマンド

コマンド	説明
repcomp	ソース側（移行元）とターゲット側（移行先）のトランスフォーマ・リポジトリの差分データを‘-patch’で指定したファイルに出力します。 出力されたファイルを repimp の‘-patch’でインポートすると、ソース側（移行元）に指定したトランスフォーマ・リポジトリと同じ内容になります。
repimp	‘-patch’で指定したファイルの差分データを、インポート先のトランスフォーマ・リポジトリに反映して、repcomp のソース側（移行元）と同じになるように更新します。 インポート先のトランスフォーマ・リポジトリは、repcomp で比較したターゲット側（移行先）のトランスフォーマ・リポジトリと同じ内容である必要があります。 差分データを使用する場合、-r を指定したインポート、オペレーションマネージャからのインポートは対応していません。

③ コマンド実行例

差分データファイルの作成

```
repcomp.sh -srn odip_db_new -td ~/backup/odiprep_old -patch /tmp/newdb.patch
```

差分データファイルのインポート

```
repimp.sh -rn odip_db_old -patch /tmp/newdb.patch
```

(2) 改定の影響範囲

本改定によって、既存定義の変更、実行結果の相違が生じることはありません。

2. PostgreSQL データ出力方法拡張

(1) 改定内容

① 出力方法の拡張

PostgreSQL へ DBMS 固有のローダを使用して出力する機能が追加されました。
ODIP トランスフォーマの設定ファイル（batchMain.conf）の WriteType に“3”または“5”のとき、出力先データベースの配置に応じて、次のいずれかの方法でデータがロードされます。

(a) PostgreSQL データベースが、ODIP トランスフォーマと同じホスト上で稼働している場合（以下、ローカル接続といいます）は、JDBC によるデータベース接続を使用して、“COPY”ステートメントを実行します。

(b) PostgreSQL データベースが、ODIP トランスフォーマと異なるホスト上で稼働している場合（以下、リモート接続といいます）は、PostgreSQL の“psql”コマンドから、“¥COPY”メタコマンドを実行します。

② DBMS 設定ファイルのオプション追加

PostgreSQL 用の DBMS 設定ファイル（postgresql.properties）に、表 2 のオプションが追加されました。DBMS 設定ファイルは、config ディレクトリにあれば有効になり、なければ既定値が適用されます。DBMS 設定ファイルは、導入時には config ディレクトリにはありませんので、必要に応じて config/jdbcsample ディレクトリから config ディレクトリへコピーしてください。

表 2　postgresql.properties の追加オプション

オプション	説明
loader.command	リモート接続で実行する“psql”コマンドのテンプレートを指定します。 <host>、<port>、<user>、<dbname>、<statement>は、実行時にそれぞれデータベースのホスト、ポート番号、データベース名、¥COPY コマンドに置換されます。 既定値： psql -h <host> -p <port> -U <user> -d <dbname> -w -c <statement>
loader.file.encoding	ロード用データファイルのエンコーディングを指定します。 既定値：UTF-8
loader.statement.local	ローカル接続で実行する“COPY”コマンドのテンプレートを指定します。 <table_name>、<column_names>、<infiles>、<charset>は、実行時にそれぞれ、出力先テーブル名、列リスト、ロードファイルパス、データファイルのエンコーディングに置換されます。 既定値： COPY <table_name> (<column_names>) FROM '<infiles>' WITH (FORMAT CSV, ENCODING '<charset>')
loader.statement.remote	リモート接続で実行する“¥COPY”コマンドのテンプレートを指定します。 <table_name>、<column_names>、<infiles>、<charset>は、実行時にそれぞれ、出力先テーブル名、列リスト、ロードファイルパス、データファイルのエンコーディングに置換されます。 既定値： ¥COPY <table_name> (<column_names>) FROM '<infiles>' WITH (FORMAT CSV, ENCODING '<charset>')
loader.type	auto、local、remote のいずれかを指定します。 auto　：ローカル接続、リモート接続を自動判定します。 local　：ローカル接続として、JDBC による COPY コマンドを実行します。 remote：リモート接続として、psql による¥COPY コマンドを実行します。 既定値：auto

③ PostgreSQL データベースへのリモート接続実行環境の留意点

・リモート接続で COPY による出力を行うには、ODIP トランスフォーマの実行環境に、PostgreSQL のクライアントソフトウェアが導入され、“psql”コマンドが実行可能

であることが前提になります。

・“psql”コマンドは、接続パスワードを引数で渡すことができません。予め Linux の場合は“.pgpass”に、Windows の場合は“pgpass.conf”に接続情報を記載してください。

(2) 改定の影響範囲

・PostgreSQL データベースを使用し、ODIP トランスフォーマの設定ファイル（batchMain.conf）の WriteType、DetailWriteType に、“3”または“5”を設定している場合には、データ出力時の動作が変わります。

・本改定によって、既存定義の変更、実行結果の相違が生じることはありません。

3. オンデマンド処理のパフォーマンス改善

(1) 改定内容

① 概要

startjob コマンドのオプションに‘-thread’が追加され、ジョブ用の JavaVM を起動せずに、ODIP トランスフォーマのメモリ内でジョブを実行することで、オンデマンド処理のパフォーマンスが改善されました（スレッドモード)。‘-thraed’を指定しない場合は従来と同じく、ジョブ用の独立した JavaVM が起動して処理を実行します。

‘-thread’を指定すると ODIP トランスフォーマサーバのメモリを使用しますので、同時実行ジョブ数を考慮して ODIP_JVM_OPTS で-Xmx を十分なサイズのメモリを指定してください。

② 改定されたコマンド

コマンド	説明
startjob	‘-thread’を指定すると、プールされたスレッドでジョブを実行し、プールされたコネクションが使用されるため、処理時間の短いジョブを繰り返し実行する場合のパフォーマンスが向上します。 ‘-thread’の指定には、‘-od’（オンデマンドモード）の指定も必須になります。また、‘-thread’を指定したジョブは batchSrv.conf の MaxJobCount の制限を受けません。スレッドモードで実行されるジョブの同時実行数を指定するには、新たに追加された batchSrv.conf のパラメータ odip.threadmode.threadpool.maxPoolSize に最大数を指定してください。

③ コマンド実行例

スレッドモードでのジョブ実行

```
startjob.sh -rn odiprep -pi 01010200010100 -pd 20190701 -od -thread
```

④ 追加されたパラメータ

‘-thread’（スレッドモード）の追加にともない、表 3 パラメータが batchSrv.conf に追加されました。スレッドモードでのみ、ジョブ実行用のスレッド、およびデータベース接続はプールされたものを使用します。

表 3 追加されたパラメータ

オプション	説明
odip.threadmode.threadpool.corePoolSize	アイドル状態であってもプール内に維持されるスレッドの数を 0 以上で指定します。0 を指定するとアイドル状態のスレッドは維持されません。 既定値：5
odip.threadmode.threadpool.maxPoolSize	使用可能なスレッド数の最大値を指定します。 odip.threadmode.threadpool.corePoolSize 以上の値を指定してください。 スレッドモードのジョブの同時実行数を制限するには、ここに最大数を指定してください。 既定値：100
odip.threadmode.threadpool.keepAliveTime	アイドル状態のスレッドが終了するまでの時間（秒）を指定します。 既定値：0（すぐに終了する）
odip.threadmode.threadpool.allowCoreThreadTimeOut	アイドル状態のスレッドがタイムアウトで終了することを許可する場合は true、アイドル状態を維持する場合は false を指定します。 既定値：false
odip.threadmode.jdbcpool.initialSize	最初にデータベース接続が発生したときに作成するコネクションの数を指定します。 既定値：10
odip.threadmode.jdbcpool.maxActive	作成可能なコネクション数の最大値を指定します。 既定値：100
odip.threadmode.jdbcpool.maxIdle	アイドル状態であっても維持されるコネクション数の最大値を指定します。 既定値：100
odip.threadmode.jdbcpool.minIdle	アイドル状態であっても維持されるコネクション数の最小値を指定します。 既定値：10
odip.threadmode.jdbcpool.maxWait	maxActive を超えたコネクションの取得要求が待機する時間（ミリ秒）を指定します。 この時間を超えてコネクションを使用できない場合はジョブがエラーで終了します。 既定値：30000（30 秒）

(2) 改定の影響範囲

本改定によって、既存定義の変更、実行結果の相違が生じることはありません。

4. リレーション属性の式/関数の対応（DB2）

(1) 改定内容

① 概要

リレーション属性の「キー関連編集」ダイアログで、式/関数(一部)を入力できるようになりました。入力した式/関数は SQL に変換され、SELECT 文の JOIN 結合の条件で使用されます。式/関数を使用する場合、他の計算式、条件式と同じように、属性は角括弧([])で囲んでください。

② 使用可能な関数とテンプレートの挿入

リレーション属性で使用できる関数は、キー項目のテキストフィールドで↓(下矢印)キーを押下することで表示され（図 1)、表示されている関数をダブルクリック、または選択して Enter キー押下でテンプレートがテキストフィールドに挿入されます。

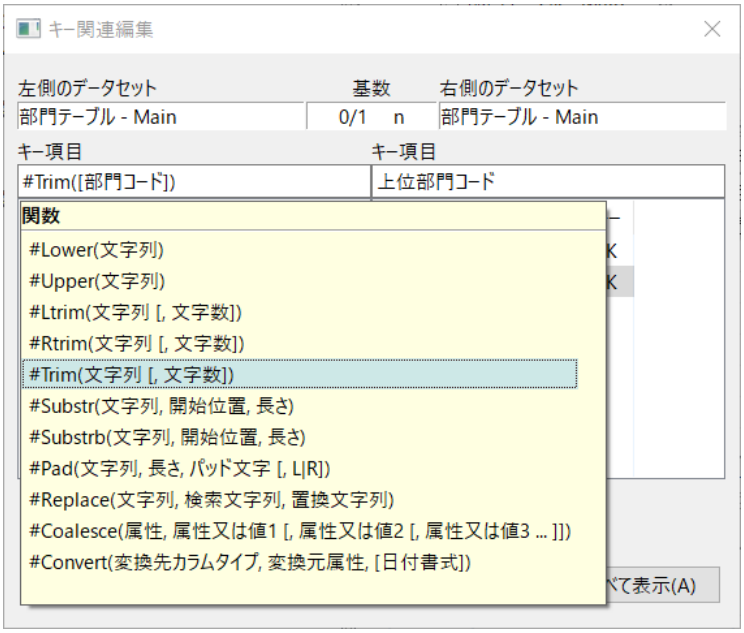

図 1 リレーション属性で使用できる関数の一覧

③ 制限

現バージョンでは、データベースが DB2 の場合のみ使用できます。その他のデータベースで実行すると、実行時にエラーが発生します。

(2) 改定の影響範囲

本改定は、再帰処理の定義に関わるもので、既存定義の変更、実行結果の相違が生じることはありません。

5. 再帰問い合わせの機能の追加（DB2）

(1) 改定内容

① 概要

入力データ定義に、再帰クエリを実行する機能が追加されました。再帰クエリを使用すると、子キー、親キーによって階層化されたテーブルに対して、階層構造をたどってデータを SELECT することができます。

② 「再帰クエリ」タブの追加

入力データのリレーション定義で、リレーションの左右に同じデータセットを定義することができるようになり、リレーション定義画面下に「再帰クエリ」タブが追加されました。リレーションの左右に同じデータセットを定義すると、再帰クエリを使用して対象データセットを SELECT します。JoinGroup 内の複数のデータセットが存在し、再帰クエリ対象のデータセットと結合する場合、再帰クエリの結果行の結合結果が返ります。

対象のリレーションを選択すると「再帰クエリ」タブの項目がアクティブになり、再帰クエリに関する情報を定義できます。各項目の内容は「表 4」になります。

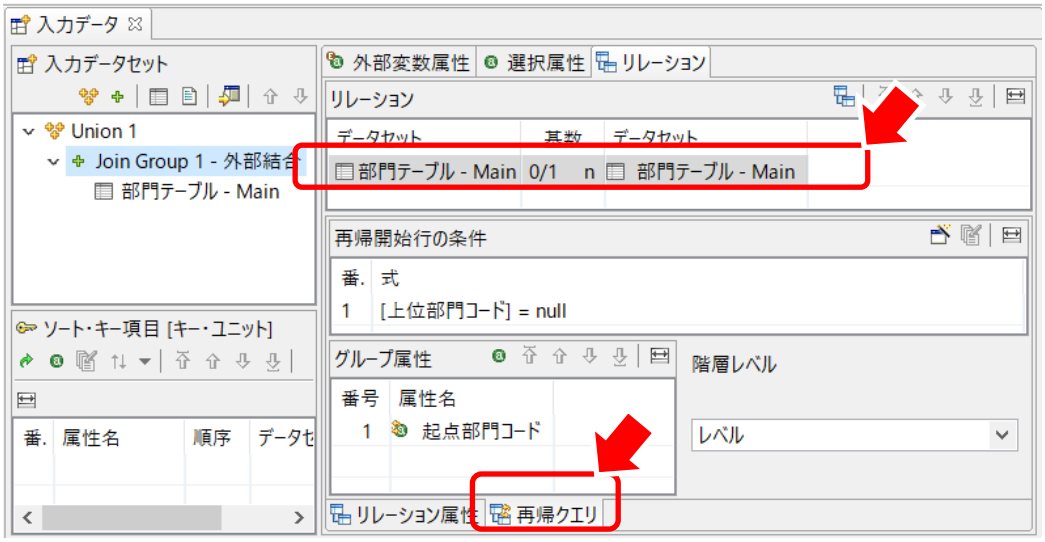

図 2 再帰クエリの定義タブ

表 4 再帰クエリタブの項目

項目	説明
再帰開始行の条件	再帰を開始する行を指定します。 例えば木構造のルート、またはリーフからのみ再帰を開始する場合に、条件を指定します。 条件式には、再帰対象データセットの「選択属性」が使用できます。 条件を指定しない場合、各ノードからの再帰結果行が返ります。
グループ属性	再帰開始行のリレーション属性の値を設定する属性を指定します。 必須ではありませんが、指定する場合はリレーション属性の定義の数と、順序に応じて型が一致する必要があります。 設定される値は、次の優先順位で決まります。 ・左右のリレーション属性のうち PK が指定されている属性 ・左右のリレーション属性のうちカラムの位置が先の属性 ・リレーションの基数が 0/1 側の属性
階層レベル	再帰の深さを設定する属性を指定します。再帰開始行は 1 です。 再帰クエリの場合、必ず指定する必要があります。

③ 制限

・データベースが DB2 の場合のみ使用できます。他のデータベースは実行時にエラーになります。

・JoinGropu の結合方法が「外部結合」の場合のみ使用できます。

・グループ属性、階層レベル属性はソートキーに指定できません。グループ属性、階層レベル属性の順序に依存した処理を行う場合、再帰クエリの定義で一度出力したデータを、別の管理単位の入力しとして処理を分ける必要があります。

・無限ループ防止のため、再帰の階層の最大は 100 になります。

(2) 定義例

以下の定義例では、図 3 の階層構造を持つ、サンプルの部門テーブル（表 5）を使用します。

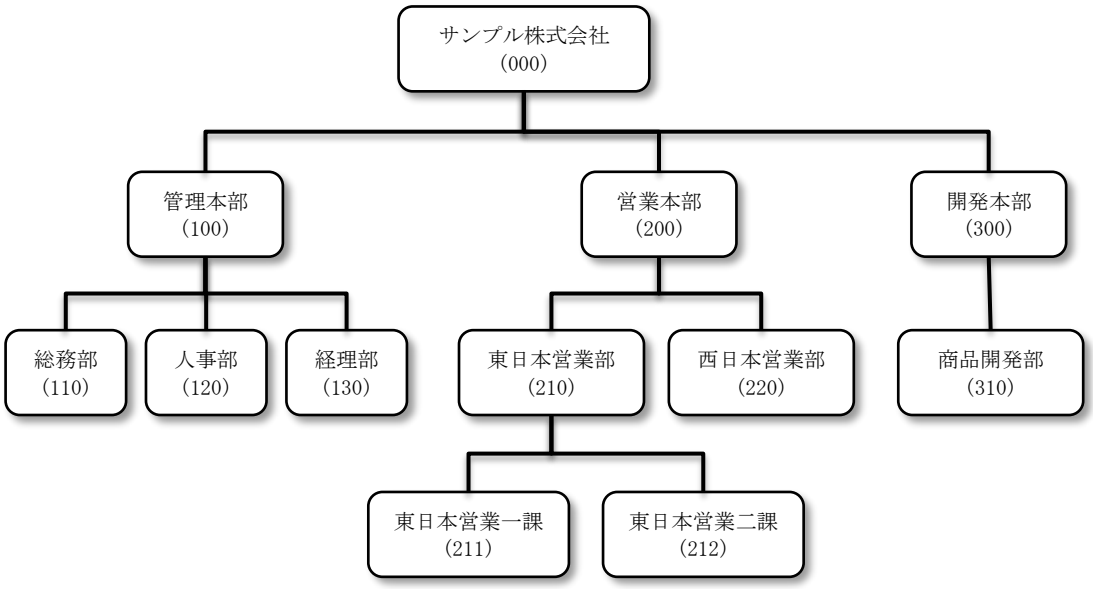

図 3 サンプル部門テーブルの階層構造

表 5 サンプル部門テーブル

部門コード	部門名	上位部門コード
000	サンプル株式会社	*null*
100	管理本部	000
110	総務部	100
120	人事部	100
130	経理部	100
200	営業本部	000
210	東日本営業部	200
211	東日本営業一課	210
212	東日本営業二課	210
220	西日本営業部	200
300	開発本部	000
310	商品開発部	300

① 定義例 1 上位から下位を検索

再帰の検索は、「基数」が 0/1 側のリレーション属性を使用して n 側のリレーション属性に対して行います。図 4 の例では、再帰開始行の条件である“[上位部門コード]＝null”の、部門コードが 000 の行から開始し、上位部門コードに 000 を持つ行を検索しています。[起点部門コード]には再帰開始行（ここではサンプル株式会社）の部門コードが設定され、[レベル]にはサンプル株式会社を起点とした階層の深さが設定されます。(表 6)

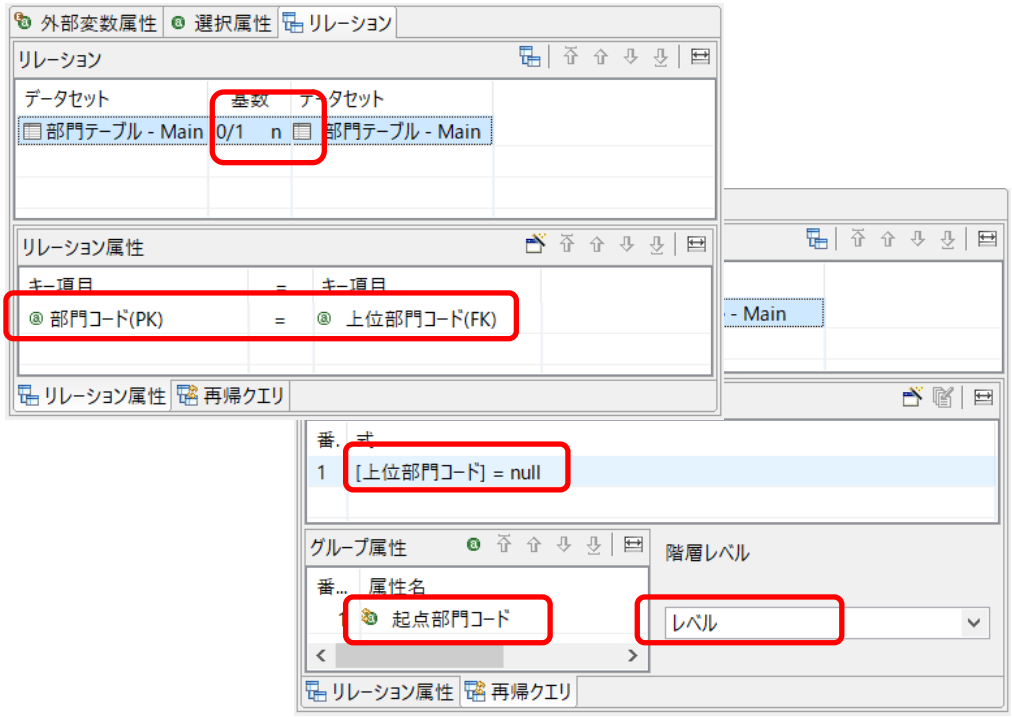

図 4 再帰クエリの定義例 1

表 6 再帰クエリの結果例 1

起点 部門コード	レベル	部門コード	部門名
000	1	000	サンプル株式会社
000	2	100	管理本部
000	2	200	営業本部
000	2	300	開発本部
000	3	110	総務部
000	3	120	人事部
000	3	130	経理部
000	3	210	東日本営業部
000	3	220	西日本営業部
000	3	310	商品開発部
000	4	211	東日本営業一課
000	4	212	東日本営業二課

② 定義例 2 下位から上位を検索

図 5 の例では、再帰開始行の条件である“[部門名]＝”東日本営業二課”” の行から開始し、基数 0/1 側の上位部門コード＝212 を部門コードに持つ行を検索しています。[起点部門コード]には再帰開始行（ここでは東日本営業二課）の部門コードが設定され、[レベル]には東日本営業二課を起点とした階層の深さが設定されます。(表 7)

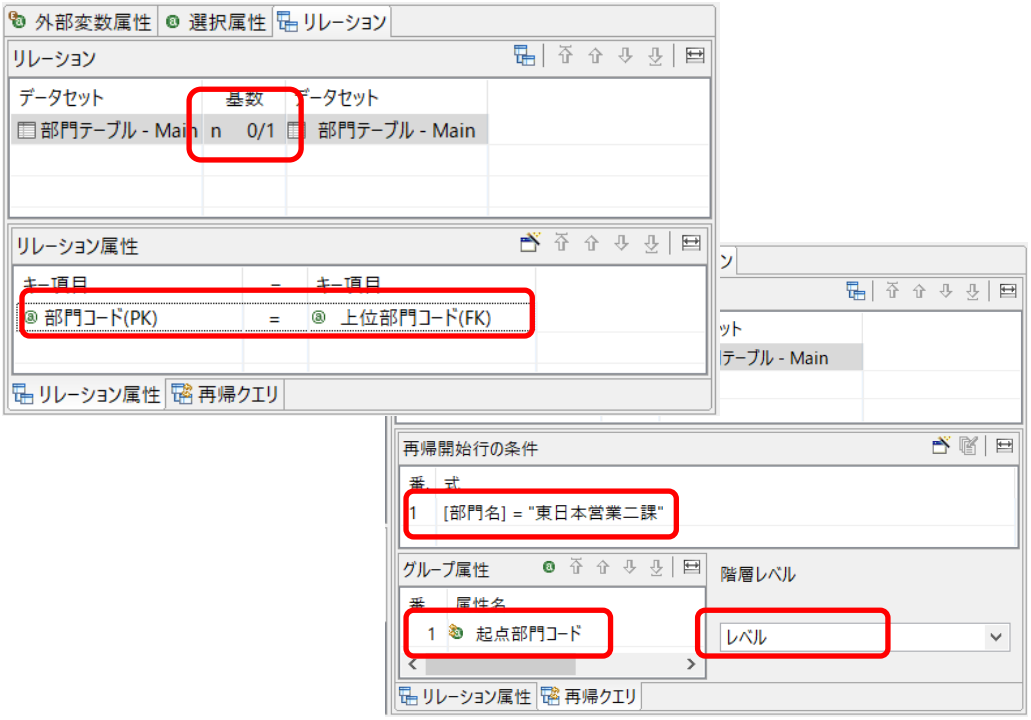

図 5 再帰クエリの定義例 2

表 7 再帰クエリの結果例 2

起点 部門コード	レベル	部門コード	部門名
212	1	212	東日本営業二課
212	2	210	東日本営業部
212	3	200	営業本部
212	4	000	サンプル株式会社

(3) 改定の影響範囲

本改定は、再帰処理の定義に関わるもので、既存定義の変更、実行結果の相違が生じることはありません。

6. その他の修正

(1) ODIP リポジトリマネージャの改定

① 改定内容

(a) ツールバーの「再表示」ボタンを押すと、プロジェクトのツリー表示にユーザ関数及びフォルダだけが表示され、他のデータタイプ、属性などが表示されなくなる問題が修正されました。また、「再表示」ボタンを押しても、更新履歴及びプロパティの画面が更新されない問題が修正されました。これらの問題は、ODIP リポジトリマネージャ v3.4 以降のバージョンで発生していました。

(b) 更新履歴ウィンドウの「変更内容」表示オプションに次の改定が行われました。

- 変更内容の詳細表示画面「相違点」ダイアログは、一つだけが開き、変更内容で選択された行の内容を表示するように変更されました。
- 「相違点」ダイアログで、入力データの選択属性の属性名などが、(Error!)と表示される問題が修正されました。
- 「相違点」の詳細表示ダイアログで、入力データの選択属性の「番号」など、定義順の表示が ODIP アドミニストレータの定義画面と異なる番号が表示される問題が修正されました。

② 改定の影響範囲

本改定は、ODIP リポジトリマネージャの画面表示だけに関わるものです。本改定によって、既存定義の変更、実行結果の相違が生じることはありません。

(2) ODIP プロセスマネジャーの改定

① 改定内容

プリファレンスの「表示」ページにおいて「編集中のファイル名をタイトルバーに表示する。(F)」オプションを有効にしても、保存先のファイル名またはリポジトリの保存先が表示されない問題が修正されました。

② 改定の影響範囲

本改定は、ODIP プロセスマネージャの画面表示だけに関わるものです。本改定によって、既存定義の変更、実行結果の相違が生じることはありません。

(3) ODIP オペレーションマネージャの改定

① 改定内容

(a) 「リカバリ管理」タブ画面で、v3.3 以降「ジョブ番号」列に表示されていた管理

単位のアイコンが、「管理単位」列に表示されるように修正されました。

(b)　「テーブル/ファイル管理」タブ及び「未使用テーブルリスト」において、フィルタを指定したときに、全角カナ文字がフィルタに一致しないと判定され、フィルタの条件に該当する名称が選択されない問題が修正されました。

(4)　ODIP トランスフォーマの改定

① 改定内容

(a)　db2.properties で定義されている LOAD コマンド構文に CLIENT オプションが追加されました。DB2 が ODIP トランスフォーマと異なるホストで稼働する場合、CLIENT オプションが必要です。DB2 が ODIP トランスフォーマと同じホストで稼働する場合、このオプションは無視されます。

(b)　処理開始時に、入力データへのデータベース接続がテーブルの数に応じて複数作成される場合がありましたが、必要な数だけ接続するように変更されました。

(c)　環境変数 ODIP_JVM_HOME に指定された JavaVM でジョブが実行されない場合がある問題が修正されました。

(d)　repcomp で表示されるビルド ID が 0000000000000000 になっている問題が修正され、正しいビルド ID が表示されるように修正されました。

(e)　#SplitCsv 関数、#SplitTsv 関数の第一引数が null の場合にエラーが発生する問題が修正され、0 が返るように変更されました。

(f)　#Substrb 関数の第二引数が 0 以下の場合にエラーが発生する問題が修正され、1（開始位置が先頭文字）として処理されるように変更されました。

(g)　オンデマンドモードのジョブ開始時に odipjob.log に出力されるメッセージの一部が、“{1}”になる問題が修正されました。

(h)　トランスフォーマ・リポジトリの構成が変更されました。旧バージョンで作成されたトランスフォーマ・リポジトリを 4.0 で使用するには、4.0 導入後に対象のトランスフォーマ・リポジトリに対して repconv.sh(.bat)を実行してください。

(i)　クロス集計、グループ集計など Interim データセットを伴う管理単位において、出力データセットのテーブル/ファイル名に変数「%n」を指定すると、「出力テーブル名に指定された変数"%n"の値が不明です。ロード・タイプが"再作成"の場合だけ変数%n は有効です」とのエラーメッセージが出力されて、処理が異常終了する問題が修正されました。

② 改定の影響範囲

本改定によって、既存定義の変更、実行結果の相違が生じることはありません。

以　上