Magic xpi 4.13

データマッパー UPSERTの使い方

マジックソフトウェア・ジャパン株式会社

OUTPERFORM THE FUTURE™

目次：データーマッパー　UPSERTの使い方

第1章

UPSERTの概要

1.1 UPSERTとは

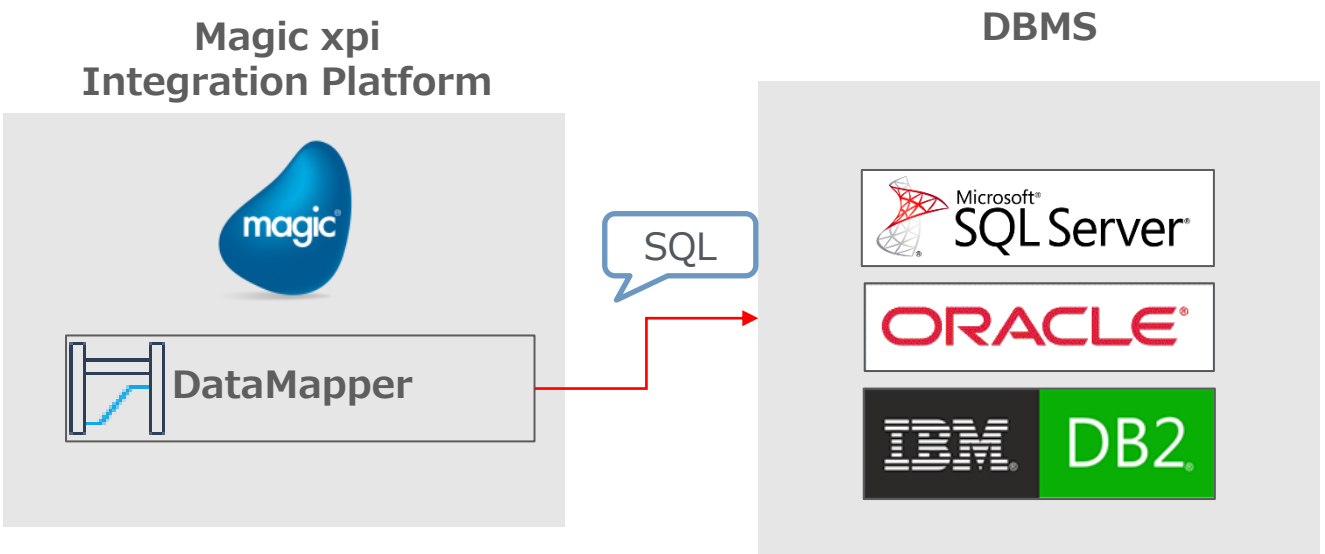
1.1 UPSERTとは

- Magic xpi のデータマッパーで**送り先にDatabaseを配置し、**
  そのプロパティのUPSERTをYesに設定すると、テーブルに対してUPSERT処理
  を行う事ができます。

　UPSERTとは、「データがあればUPDATE、なければINSERTする」という処理です。
各DBMSにより、UPSERT用のSQLが用意されており、そのSQLを
Magic xpiのデータマッパーで　発行することで、UPSERT処理が実現されます。

1.1 UPSERTとは

- UPSERTとは、データの新規挿入（INSERT）ができれば挿入を行い、
  新規挿入ができなければ更新（UPDATE）を行います。



1.2
サポートする
データベース

1.2 サポートするデータベース

- DatabaseTriggerがサポートするデータベースは下記の通りです。

DBMS	バージョン
MS-SQL	2008, 2008R2, 2012, 2014, 2016, 2017,2019
Oracle	12c, 18c, 19c
DB2/400	V7R1, V7R2, V7R3, V7R4

1.3
UPSERTの仕組み

1.3 UPSERTの仕組み

- 各DBMSで用意されている、UPSERT用のSQL文を組み立て、マップで線を引くことで、UPSERT用のSQLが発行されるように組み込みます。

- Microsoft SQLServer 、 Oracle 、 DB2/400の場合、MERGE文を使います。

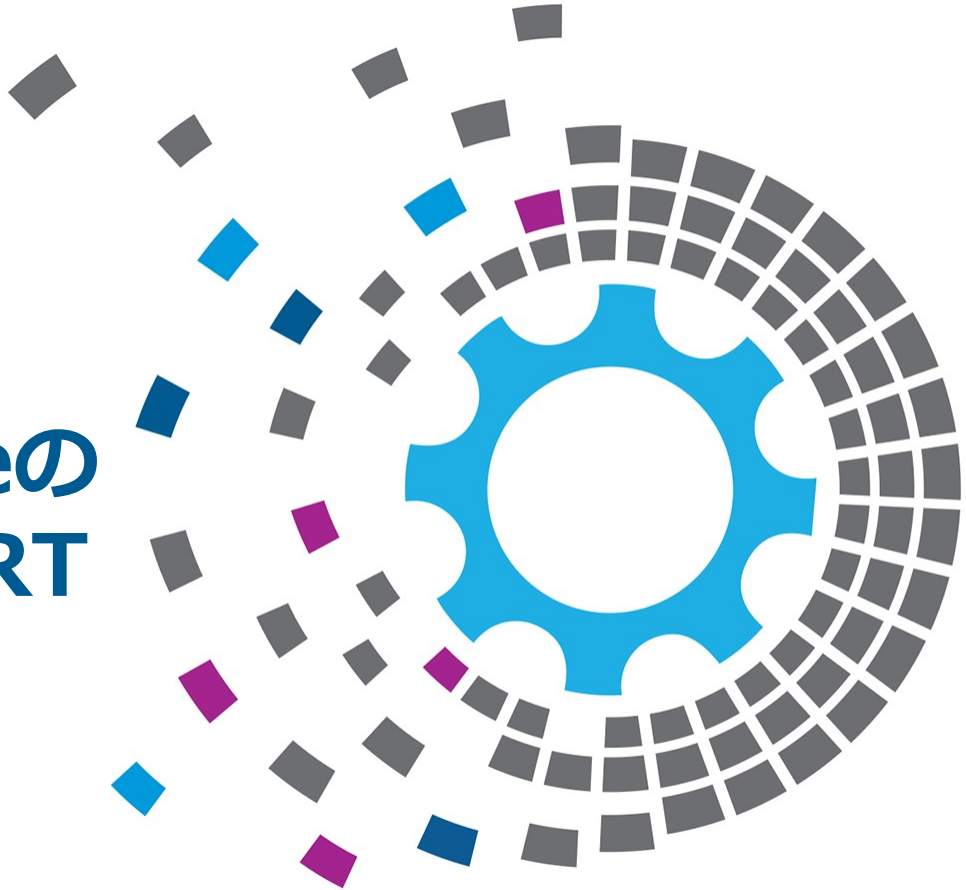
- MERGE文の基本構文：

```
MERGE INTO 主表 USING 副表 ON (条件)
  WHEN MATCHED THEN
    UPDATE SET 列1 = 値1 [, 列2 = 値2 ...]
  WHEN NOT MATCHED THEN
    INSERT (列1 [, 列2 ...]) VALUES (値1 [, 値2 ...])
```

第2章

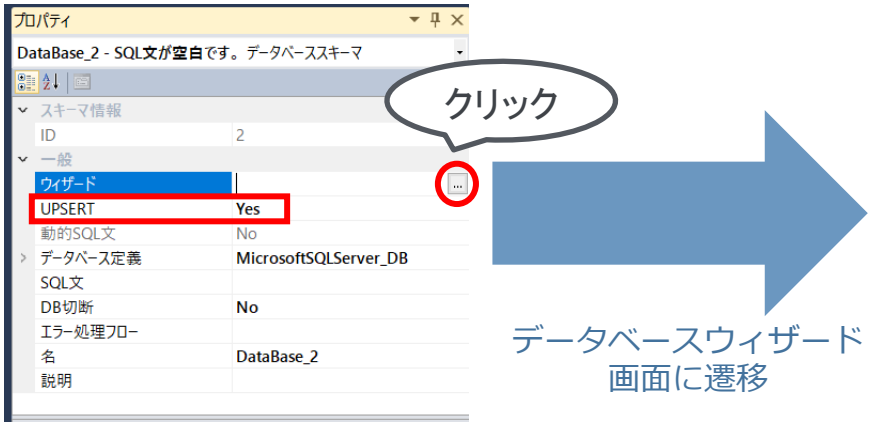
UPSERTの使い方

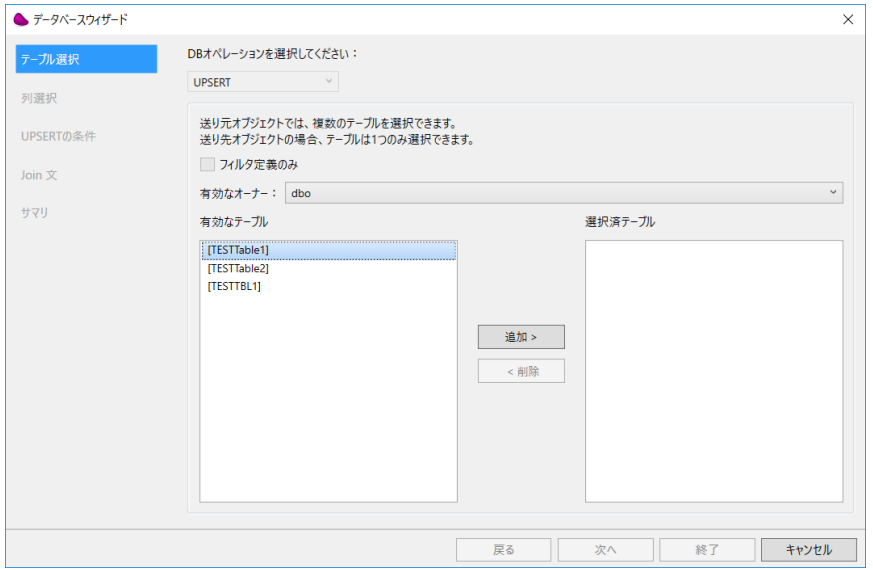
2.1
データマッパー 送り先 DataBaseの プロパティUPSERT



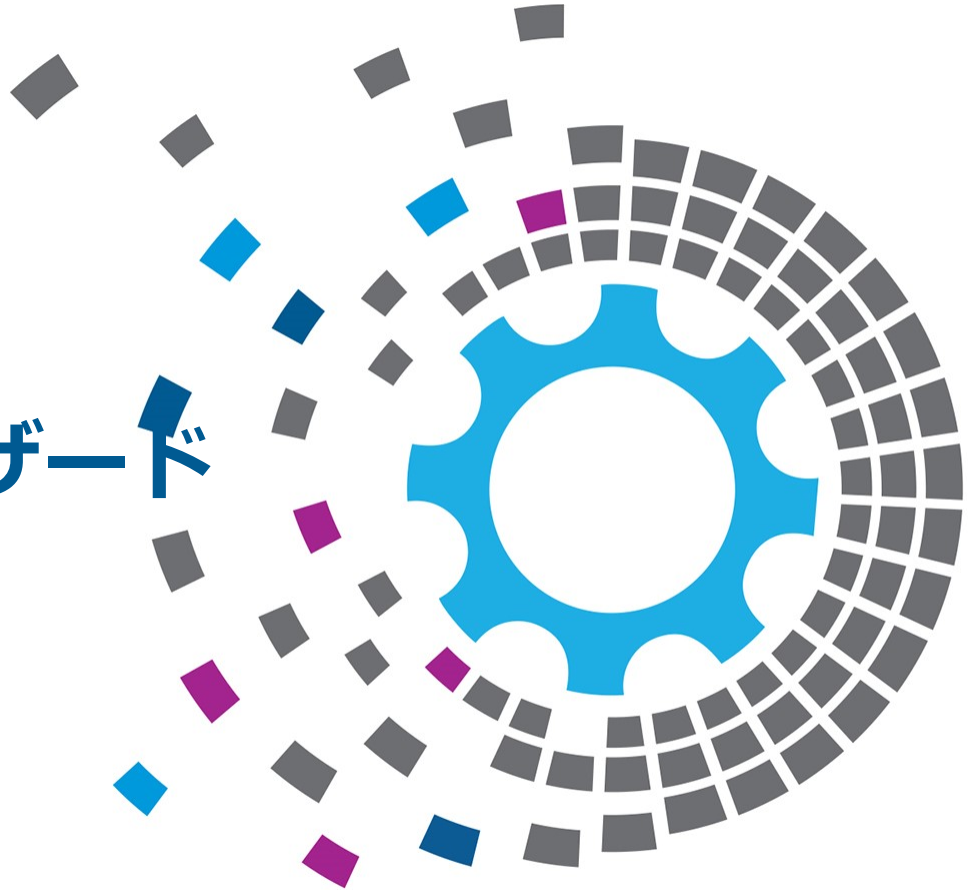
2.1 データマッパー：送り先 DataBaseのプロパティ UPSERT

- データマッパーの送り先にDatabaseを配置し、そのプロパティのUPSERTをYesに変更します。
- データベースウィザード機能を利用して、SQL文を生成します。
  （DBMSに則したUPSERT用SQL文の生成を支援してくれます）





2.2
データベースウィザード



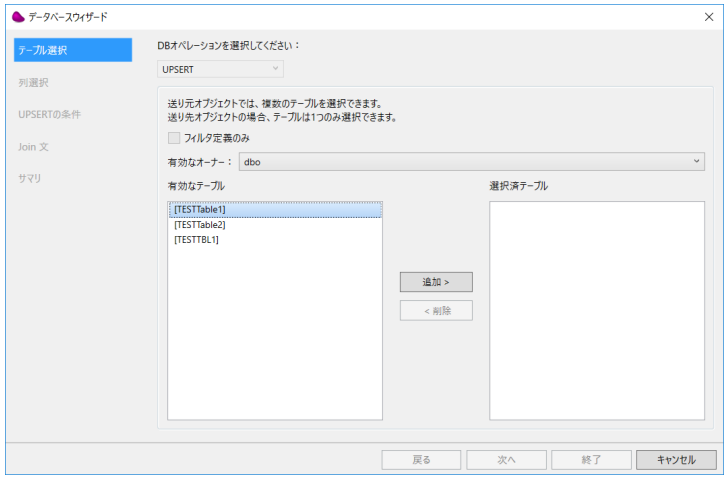
2.2 データベース ウィザード

- データベース ウィザード画面に従って進めると、各DBMSに則したUPSERT用のSQL文が生成されます。

  次ページから、データベースウィザードの操作画面を説明します。

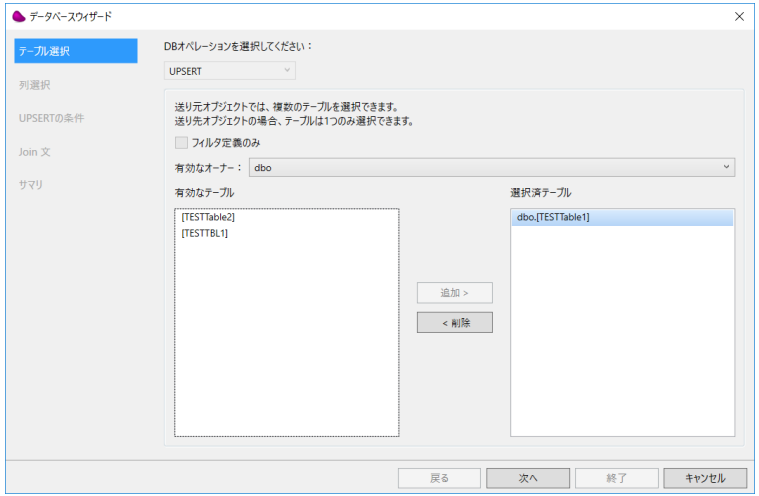
2.2 データベースウィザード（1）

- データベースがMicrosoft SQLServerの場合の操作画面例：



中央左の「有効なテーブル」のリストボックスから、目的のテーブルをクリックして選択し、[追加]をクリックします。 [次へ]をクリックする。

次へ

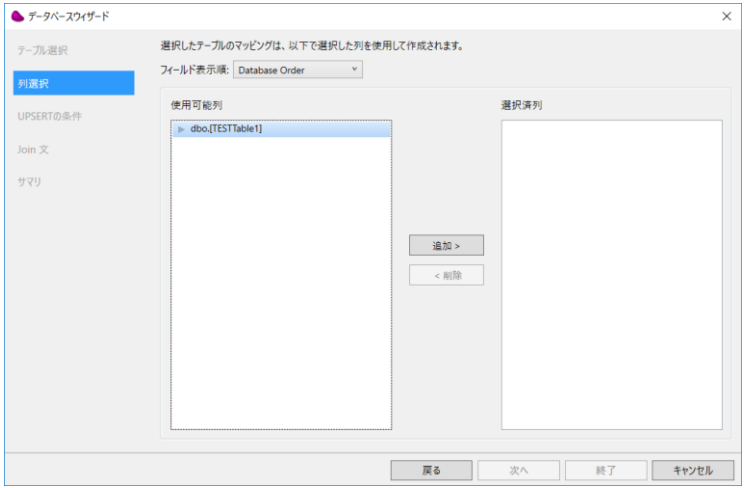


選択が完了したら、[次へ]をクリックします。

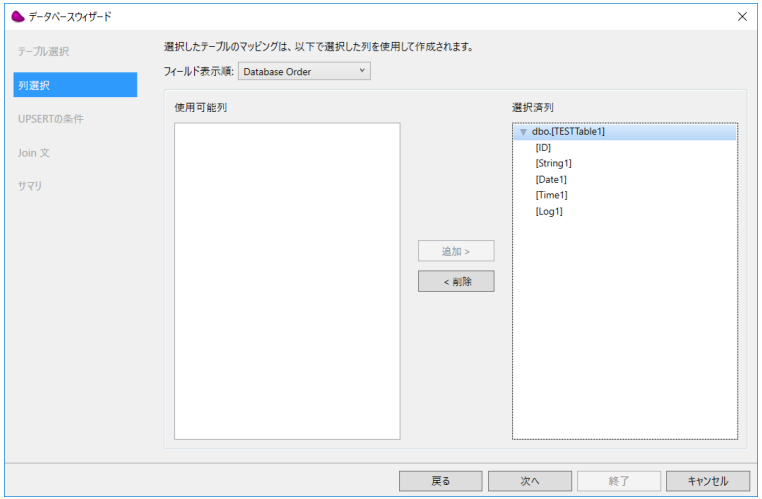
次へ

2.2 データベースウィザード(2)

- データベースがMicrosoft SQLServerの場合の操作画面例:



次へ



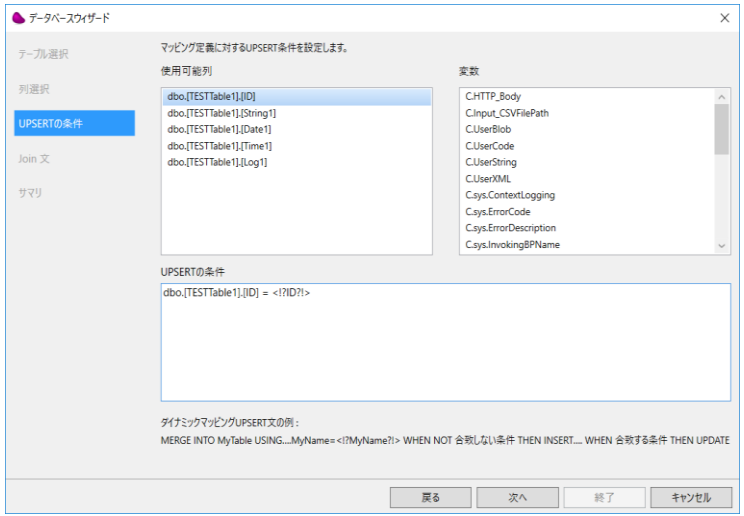
使用可能列のリストボックスのテーブル横の▼をクリックして展開し、必要なカラムをクリックして選択し、[追加]ボタンをクリックします。

全て追加する場合は、テーブルを選択して[追加]をクリックします。

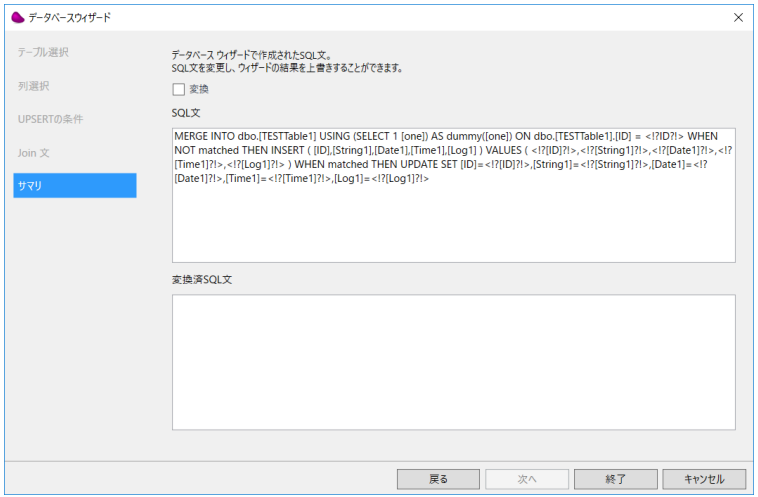
列（カラム）の選択が完了したら、[次へ]をクリックします。

2.2 データベースウィザード(3)

- データベースがMicrosoft SQLServerの場合の操作画面例：



次へ



UPSERTの条件式を設定します。
下部のUPSERT条件欄のカーソルの位置を確認し、使用可能列のリストボックスから、テーブルの列をクリックして選択し、[Enter]キーを押下すると、欄に列が入力補佐されます。
列をダブルクリックしても、欄に入力補佐されます。

データベースウィザードにより生成されたSQL文の確認・編集画面に遷移します。
SQL文の最終的な編集を行うことが出来ます。

= の右横に、<!?カラム名?!>と入力します。
この記入により、マッピング画面でノードとして表示されます。
条件式の入力が完了したら、[次へ]をクリックします。

2.3 データベースウィザードで生成されたSQL文の修正

2.3 データベースウィザードで生成されたSQL文の修正

- データベースウィザードで生成されたSQL文について（Microsoft SQLServerの場合の例）
  MERGE文の構文との対比

生成されたSQL文

```
MERGE INTO dbo.[TESTTable1] USING (SELECT 1 [one]) AS
dummy([one]) ON dbo.[TESTTable1].[ID] = <!?ID?!>
 WHEN NOT matched THEN
 INSERT ( [ID],[String1],[Date1],[Time1],[Log1] ) VALUES
( <!?[ID]?!>,<!?[String1]?!>,<!?[Date1]?!>,<!?[Time1]?!>,
<!?[Log1]?!> )
 WHEN matched THEN
 UPDATE SET
[ID]=<!?[ID]?!>,[String1]=<!?[String1]?!>,[Date1]=<!?[Dat
e1]?!>,[Time1]=<!?[Time1]?!>,[Log1]=<!?[Log1]?!>
```

MERGE文の構文

```
MERGE INTO 主表 USING 副表 ON (条件)
  WHEN NOT MATCHED THEN
    INSERT (列1 [, 列2 ...]) VALUES (値1 [, 値2 ...])
  WHEN MATCHED THEN
    UPDATE SET 列1 = 値1 [, 列2 = 値2 ...]
```

副表はダミーテーブルを使用。
（送り元のタイプがFlatFileや他のデータベースのテーブルなどに対処が可能）
条件はON句で主表の主キー列との条件式を構成する。（<!?[カラム名]?!>を記述する）
※最終的にこの構造パターンに組み立てるのは、Microsoft SQLServerも、Oracleも、DB2/400も同様です。

2.3 データベースウィザードで生成されたSQL文の修正

- SQL文の修正（Microsoft SQLServerの場合）

```
MERGE INTO [TESTTable1] USING (SELECT 1 [one]) AS dummy([one]) ON [TESTTable1].[ID] = <!?ID?!> WHEN NOT matched THEN
 INSERT ( [ID],[String1],[Date1],[Time1],[Log1] ) VALUES
( <!?[ID]?!>,<!?[String1]?!>,<!?[Date1]?!>,<!?[Time1]?!>,<!?[Log1]?!> )
 WHEN matched THEN
 UPDATE SET
[String1]=<!?[String1]?!>,[Date1]=<!?[Date1]?!>,[Time1]=<!?[Time1]?!>,[Log1]=<!?[Log1]?!>
```

UPDATE文のSET箇所の主キーカラムの削除（主キーを更新しない：[ID]）。

必要に応じて、スキーマ名の省略（「dbo.」の削除）。

　省略すると、一般的にはログオンユーザと同じスキーマまたはデフォルトのスキーマが適用されます。

2.3 データベースウィザードで生成されたSQL文の修正

- SQL文の修正（Oracleの場合）

```
MERGE INTO XPI."TESTTABLE1" USING SYS.dual ON ( XPI."TESTTABLE1"."ID"= <!?ID?!> )
 WHEN MATCHED THEN
 UPDATE SET "STR1"=<!?"STR1"?!>,"DATE1"=<!?"DATE1"?!>,"LOG1"=<!?"LOG1"?!>
 WHEN NOT MATCHED THEN
 INSERT ( "ID","STR1","DATE1","LOG1" ) VALUES
( <!?"ID"?!>,<!?"STR1"?!>,<!?"DATE1"?!>,<!?"LOG1"?!> )
```

UPDATE文のSET箇所の主キーカラムの削除（主キーを更新しない：”ID”）。

必要に応じて、スキーマ名の省略（「XPI.」の削除 / SYS.dual の「SYS.」も削除（省略）してもOK）。

　省略すると、一般的にはログオンユーザと同じスキーマまたはデフォルトのスキーマが適用されます。

2.3 データベースウィザードで生成されたSQL文の修正

- SQL文の修正（DB2/400の場合）

  データベースウィザードで生成されるSQL文の構文パターン

  MERGE INTO <ライブラリ名>.<テーブル名（ファイル名）> AS tgt
  USING (SELECT [カラム] , [カラム …] FROM <ライブラリ名>.<テーブル名（ファイル名）>) AS src
  ON (tgt.<主キーとなるカラム名> = <!?[カラム名]?!>)
  WHEN MATCHED THEN
  UPDATE SET [カラム名]=<!?[カラム名]?!> [, カラム名=<!?カラム名?!> …]
  WHEN NOT MATCHED THEN
  INSERT ([カラム名] [, カラム名 …]) VALUES (<!?[カラム名]?!> [,<!?カラム名?!> …])

  USING の副表と主表が同じであるため、DB2/400では、実行時にエラーとなることが確認されています。
  USINGはダミーテーブルを利用するよう修正します。
  修正例１： VALUES('DUMMY’)
  修正例２： SELECT * FROM (VALUES(‘DUMMY’)) AS T1
  （注：SELECTでVALUESを扱う場合、AS句で別名を指定する必要があります。指定しないとエラーとなります。）

  UPDATE文のSET箇所の主キーカラムの削除（主キーを更新しない）。

2.3 データベースウィザードで生成されたSQL文の修正

- 修正する場合のその他の留意点
  - ON句の条件で使用するカラムは、主キーはユニークキーまたはプライムキーであること。
  - 主キーが複合主キー（複数のカラムの組み合わせで構成）の場合は、ON句にすべてのカラムの条件式を記述する

    （[複合主キーカラム１] = <!?[カラム1]?!> , [複合主キーカラム2] = <!?[カラム2]?!> [, …]）

THANK YOU!

magic
OUTPERFORM THE FUTURE™