Magic uniPaaS V1Plus
DLLMaker 利用ガイド

目次

1. はじめに

uniPaaS DLLMaker は、uniPaaS で作成した Btrieve ファイルを、SQL テーブルとしてアクセスできるようにするためのユーティリティです。
本書では、DDLMaker の基本概念、セットアップ方法および利用方法を説明します。

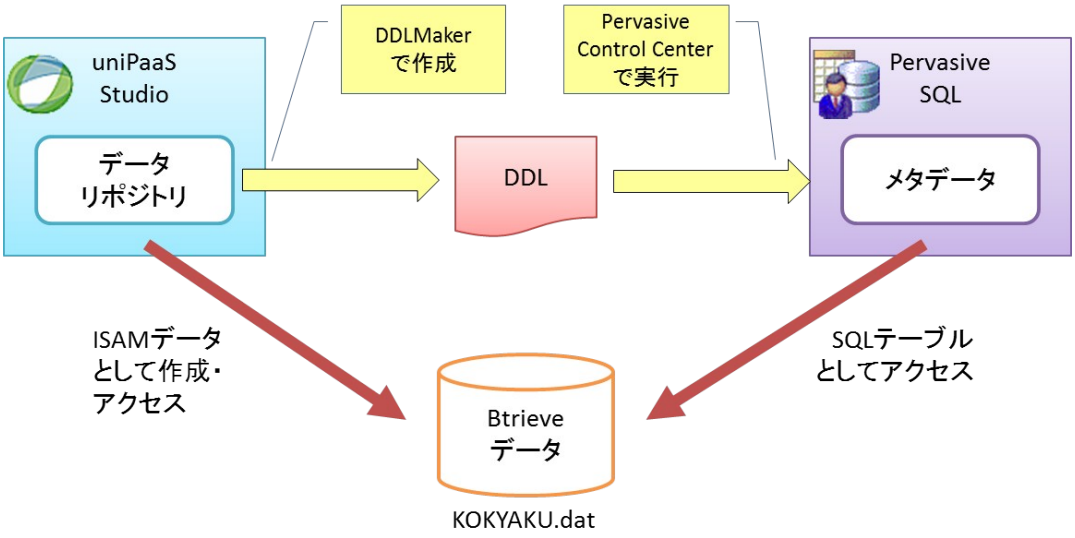
従来、Btrieve データを SQL テーブルとして扱うためには、DDF ユーティリティ (Magic eDeveloper V9Plus まで)、あるいは DDFMaker (Magic uniPaaS　V1Plus (Ver1.8SP1) まで) として提供されていたユーティリティを利用して行っていました。このユーティリティは、Magic 開発版 (Studio) のテーブルリポジトリを読み込み、Pervasive.SQL のメタデータ (DDF) を作成します。Pervasive.SQL はこの DDF を使って、リレーショナルテーブルとしてファイルを認識していました。
これらのユーティリティは DDF を直接操作するものですが、Pervasive 社の情報によると、DDF を直接操作すると、DDF の非公開データなどが作成されないため、正しい DDF の構造にならない可能性があり、現在の Pervasive.SQL では推奨されていません。代わりに、Pervasive 社は、IN DICTIONARY 句を使った CREATE TABLE/INDEX 文により、メタデータを作成する方法を推奨しています。
このような背景から、新しいユーティリティとして、DDLMaker を提供することになりました。

- 本ユーティリティは、Btrieve ファイルを SQL テーブルとしてアクセスするための補助ツールとして提供するものです。細心の注意を払って確認を行っておりますが、すべてのケースについて常に正しい結果を出力することを保証するものではありません。本ユーティリティを利用した結果については、MSJ/MSE 社は一切の責任を負いません。
- 本ユーティリティは出力として DDL 文を作成し、実際の Btrieve データに対しては一切変更を行いません。開発者は、まず、作成された DDL 文を良く確認してから、Pervasive Control Center などで実行するようにしてください。
- Btrieve から Pervasive.SQL への移行は、操作を誤ると、データ破損などの危険を伴う可能性がある作業です。移行を行う前に、Btrieve ファイルのバックアップを取っておいてください。また、Btrieve ファイル構造をよく理解した上で慎重に行ってください。
- Pervasive.SQL と Btrieve との基本仕様の違いにより、Btrieve ファイルから Pervasive.SQL に移行できない場合があります。また、uniPaaS 独自のデータ型では、Pervasive.SQL で BINARY データとしてアクセスできても、本来のデータの意味を正しく認識できない場合もあります。
- Btrieve ファイルの物理的構造と、CREATE TABLE (下記)での定義とが正しく対応していないと、定義と実際のデータ構造との乖離が起こり、正しくデータをアクセスできないとか、最悪、データ破損や動作の異常を引き起こす可能性があります。

Btrieve ファイルと、Pervasive PSQL テーブルとのデータ型の対応等に関する技術的詳細は、リファレンスヘルプの「Magic uniPaaS で使用するデータベース→ Pervasive (SQL) データベース」以下に詳説してありますので、本ユーティリティを利用する前に、必ず一読しておいてください。

2. DDLMaker 利用の流れ

DDLMaker を利用のおおまかな流れは、以下のようになります。



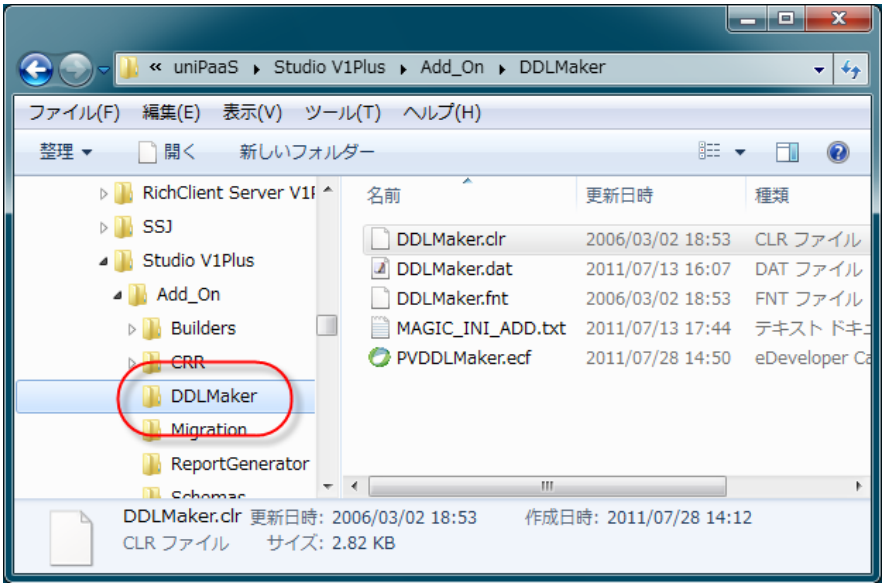
ここでは、uniPaaS Studio のデータリポジトリで、KOKYAKU.dat という名前の Btrieve ファイルが定義されていて、これを SQL テーブルとしてアクセスできるようにする手順を例にしています。

#	説明	詳細説明
1.	uniPaaS Studio で DDLMaker を利用して、このテーブルに対する DDL 文 (CREATE TABLE 文および CREATE INDEX 文) を作成します。この DDL は、指定したディレクトリにテキストデータとして格納されます。	第 4 章 DDLMaker の使い方
2.	Pervasive Control Center (以下、PCC と略) を使って、データベースを作成します。このとき、データベースディレクトリとしては、Btrive ファイル KOKYAKU.dat のあるディレクトリを指定します。	第 5 章 データベースの作成
3.	PCC で、作成した DDL 文を実行させます。	第 6 章 DDL 文の実行
4.	Pervasive PSQL のユーティリティ DDF Builder で、テーブルの確認をします (任意)。	第 7 章 DDFBuilder によるテーブルの確認

3. DDLMaker の設定

DDLMaker は、uniPaaS Studio 製品のインストーラが自動的にセットアップしますが、設定を確認したい場合には、以下のようにしてください。

1. uniPaaS のディレクトリの下に、Add_On¥DDLMaker ディレクトリが存在し、以下のようなファイルがあることを確認してください。
   ※ 日付等はバージョンにより異なることがあります。



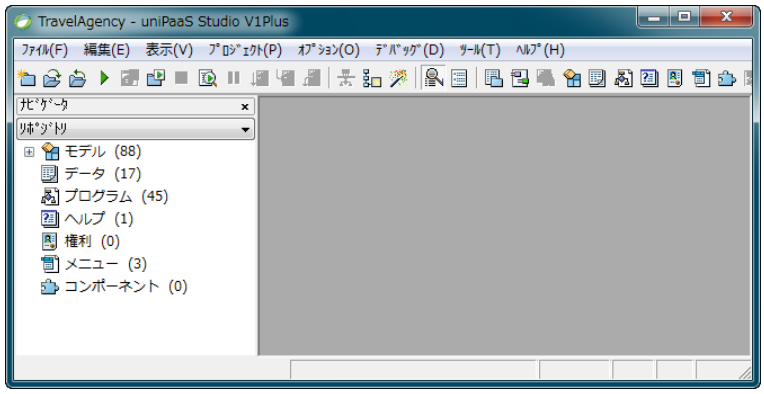
2. unipolar Studio ディレクトリの MAGIC.INI ファイルに、以下の記述があることを確認してください。
   ※ この内容は、Add_On¥DDLMaker ディレクトリの MAGIC_INI_ADD.txt からコピーすることができます。

```
[TOOLS_MENU]
DDLBDR = A,DDLMaker(&D),,Add_On¥DDLMaker¥PVDDLMaker.ecf,,,Add_On¥DDLMaker¥D+
DLMaker_suf.opr,ImageFor = B ToolNumber = 60 ToolGroup = 1 ToolTip = +
"ﾃﾞｰﾀ辞書を作成します."
```

4. DDLMaker の使い方

ここでは、Studio に添付のサンプルプロジェクト「TravelAgency」で使う Btrieve データを例にとって、DDLMaker を使って、SQL テーブルとして登録する方法を説明します。

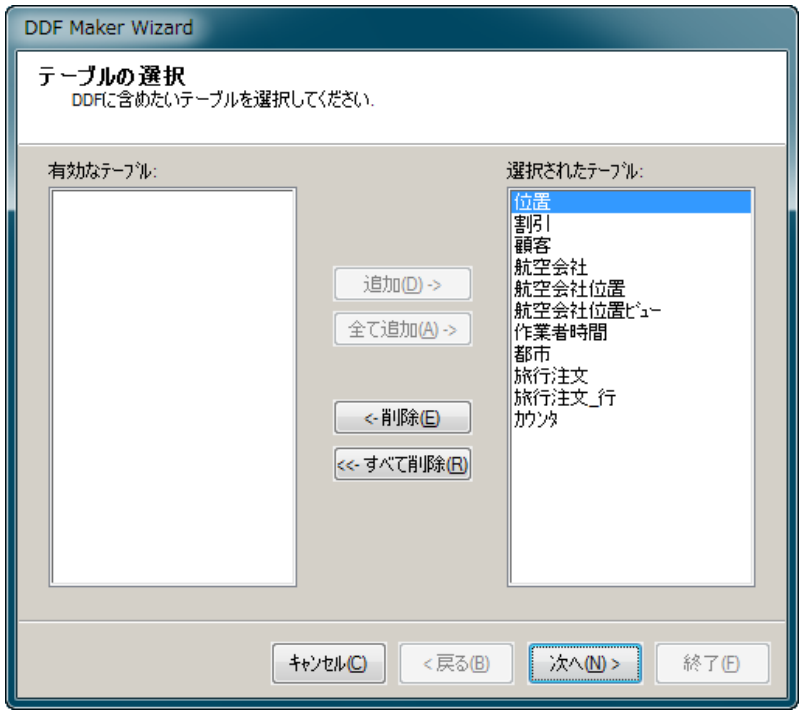
プロジェクトを開きます。



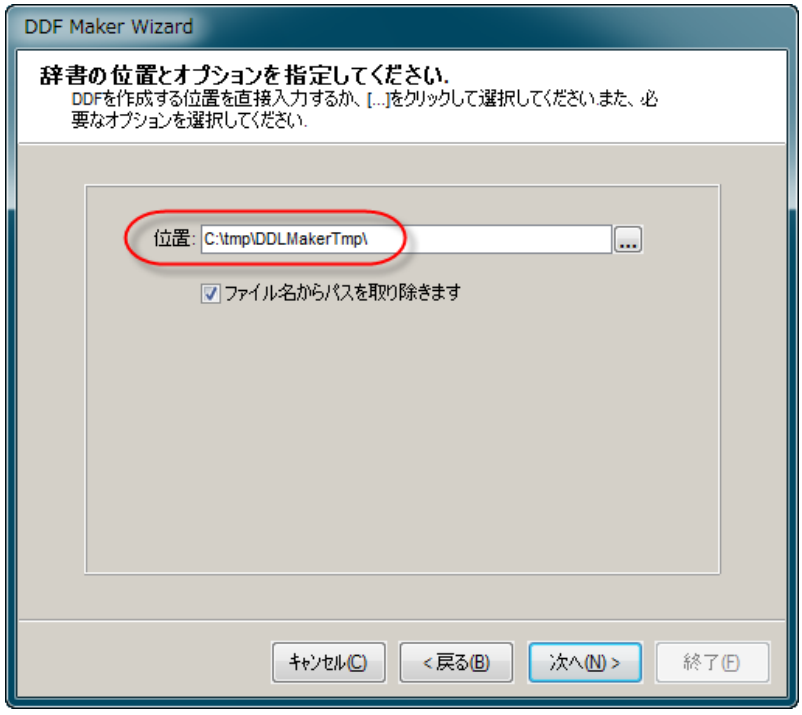
メニュー [ツール(T) → DDLMaker(D)] を選択します。

ウィザードが開始されます。
[次へ] を押してください。

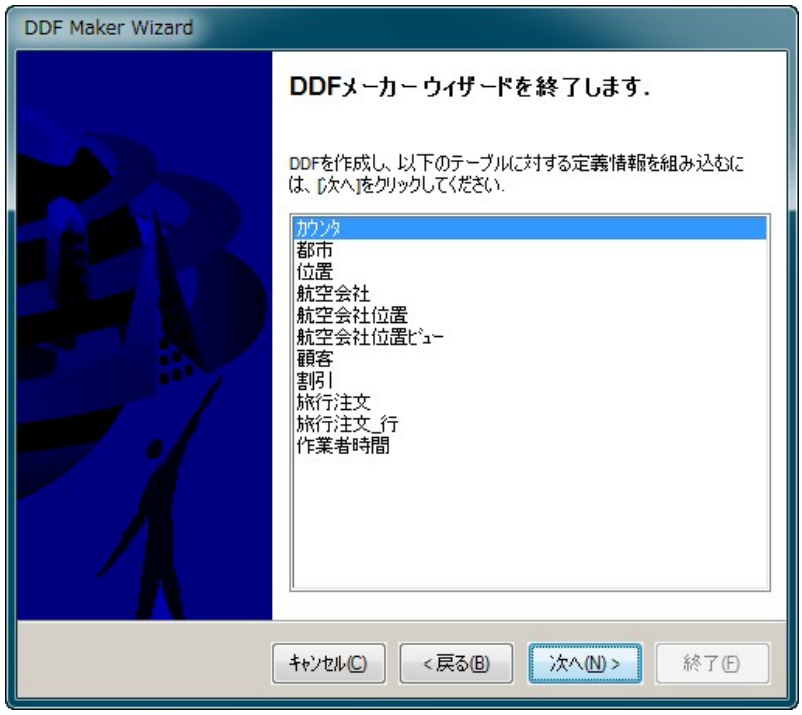
「テーブルの選択」画面が出ます。
DDL を作成したいテーブルを選択してください。
個々のテーブルを選択する場合には、テーブルを選んで[追加(D)]ボタンを押し、すべてのテーブルを選択する場合には[全て追加(A)] ボタンを押します。
右図は、すべて選択した状態を表示しています。
[次へ] ボタンを押します。



DDL を格納するファイルを作成するディレクトリ名を「位置」欄に指定します。
このディレクトリは、任意の一時ディレクトリで構いません。

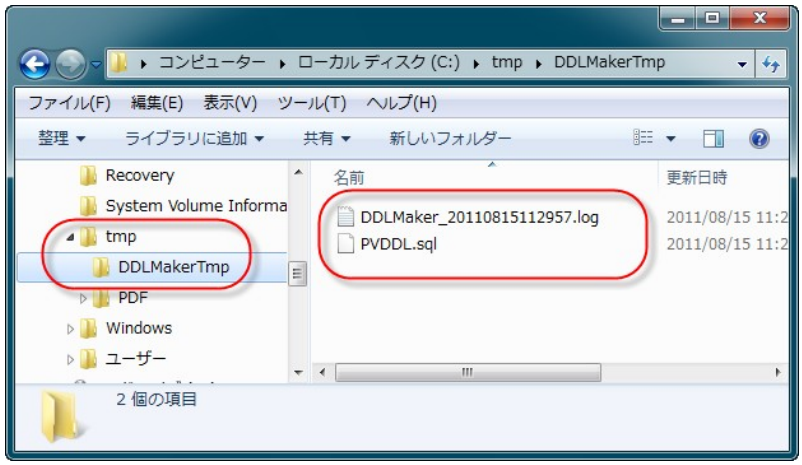


確認ダイアログが出てきますので、テーブル名を確認します。
これで良ければ [次へ] ボタンを押すと、DDL が作成されます。



以上でウィザードが終了します。

「位置」に指定したディレクトリを開くと、右図のようなファイルが作成されています。

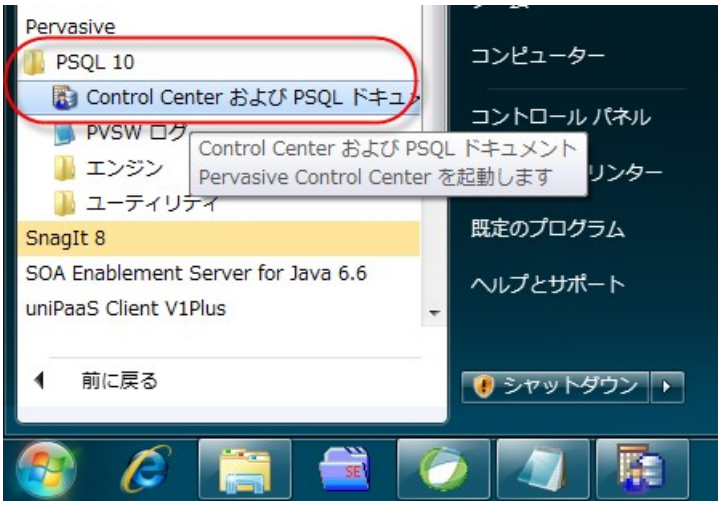


作成された DDL 文は、PVDDL.sql という名前のファイルに、テキストデータとして格納されています。
DDLMaker_(日付)(時刻).log という名前のファイルもありませすが、これには、ウィザードが処理を行なったログが記録されています。エラーや警告等が記録されていますので、確認してください。
DDLMaker の出力ファイルの詳しい内容については、第 9 章 DDLMaker の出力 を参照してください。

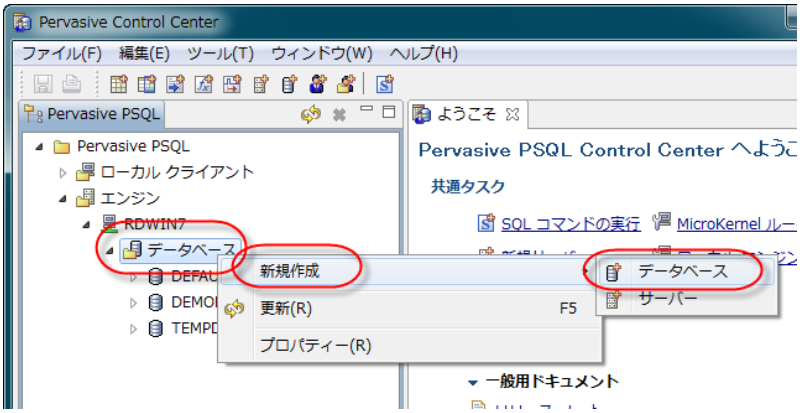
5. データベースの作成

Pervasive.SQL のデータベースの定義は、Pervasive Control Center で行います。

Windows のスタートメニューから、「Pervasive → PSQL 10 → Control Center および PSQL ドキュメント」を選択します。
※ Pervasive のバージョンによって、メニュー名が異なります。

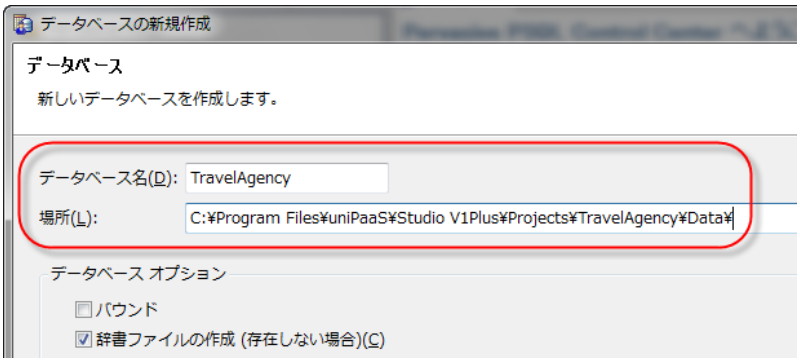


エンジンの「データベース」から、「新規作成 → データベース」メニューを選びます。



「データベース名」には、適当な名前を指定します。
ここでは、TravelAgency という名前にしています。

「場所」には、Btrieve ファイルのあるディレクトリを指定します。
他のパラメータは、そのままにします。
「終了」ボタンを押すと、データベースが作成されます。



以上で、データベースが作成されました。

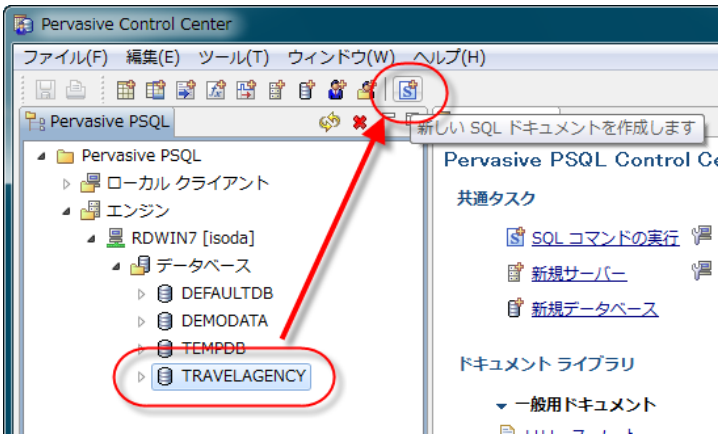
データベース名について：

- データベース名は、Pervasive PSQL のデータベース名の規約にのっとっていなければなりません。Pervasive. PSQL に添付のドキュメント「上級者用ドキュメント > Advanced Operations Guide > Pervasive PSQL データベース」を参照してください。
  一般には、特殊文字を使わず、20 文字以内半角英数字の名前とするのが無難です。
- データベース名では、大文字・小文字を区別しません。
- デフォルトでは、指定したデータベース名と同じ名前で、ODBC の データソース名 (DSN) が作成されます。後に uniPaaS から SQL テーブルとしてアクセスする場合に、「データベース」テーブルに指定する名前となります。

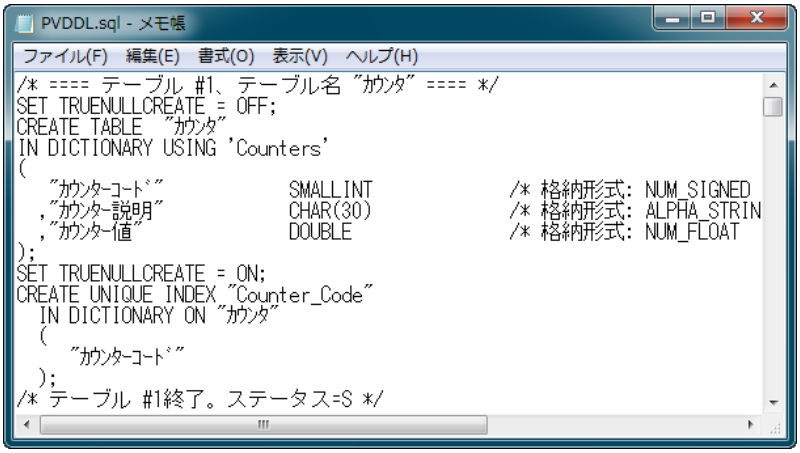
6. DDL 文の実行

次には、作成したデータベース上で、DDLMaker で作成した DDL 文を実行して、テーブルを定義します。

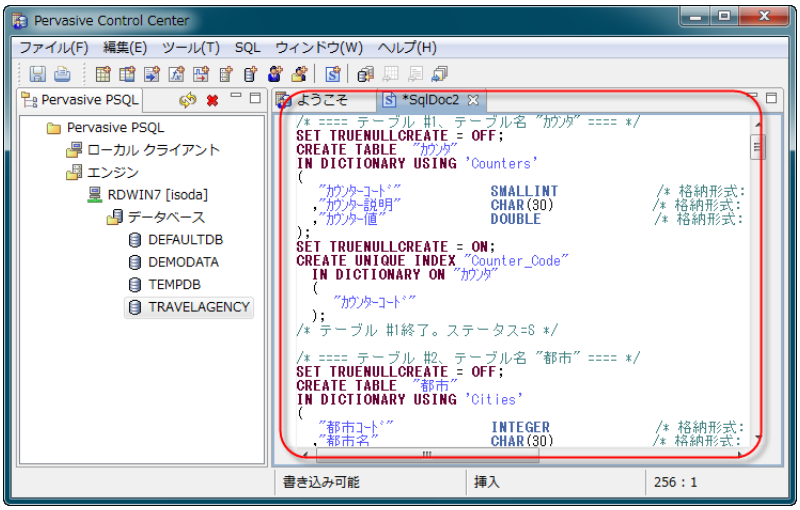
作成されたデータベースを選んで、「新しい SQL ドキュメントを作成」ボタンを押します。



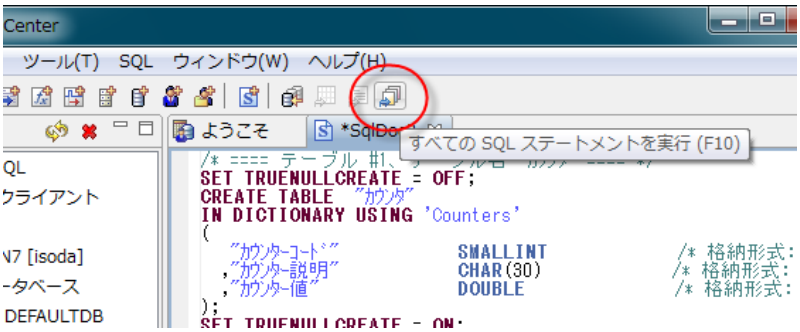
先に作成した PVDDL.sql ファイルをメモ帳などで開きます。



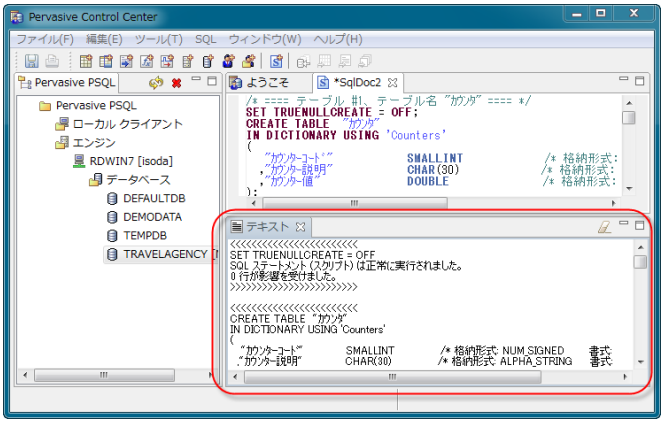
PVDDL.sql の内容を SQL ドキュメント画面にコピー＆ペーストします。



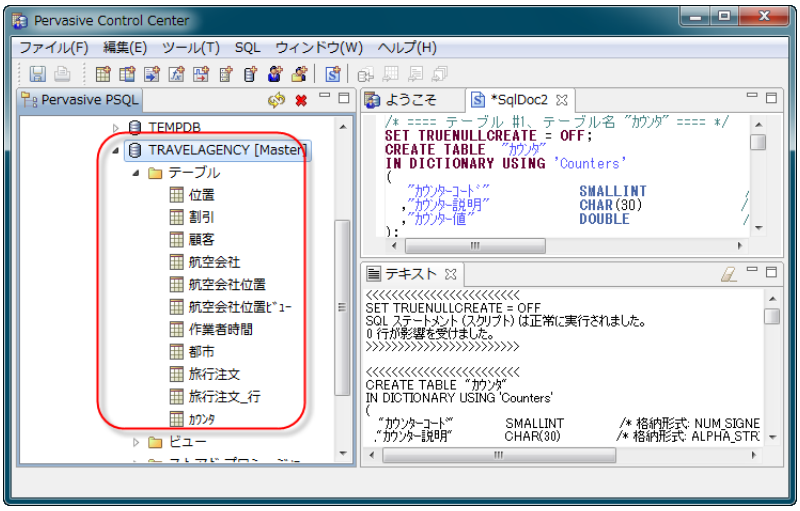
「すべての SQL ステートメントを実行 (F10)」 ボタンを押します。
コピーした DDL 文が実行されます。



実行結果は、「テキスト」画面に表示されますので、エラー等がないことを確認してください。



また、「テーブル」を開いて、必要とするテーブルがすべて定義されたかを確認してください。



以上で、データベースに SQL テーブルが定義されました。

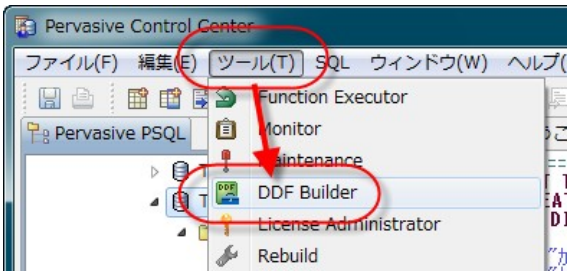
7. DDFBuilder によるテーブルの確認

Pervasive.SQL テーブルが正しく定義されたかを確認するには、Pervasive PSQL に付属の DDF Builder を使います。
この作業は必須ではありませんが、定義と実際のデータ構造とが合致しているかを確認することは、データ整合性を確保する上で大変重要ですので、慎重にチェックすることをお勧めします。

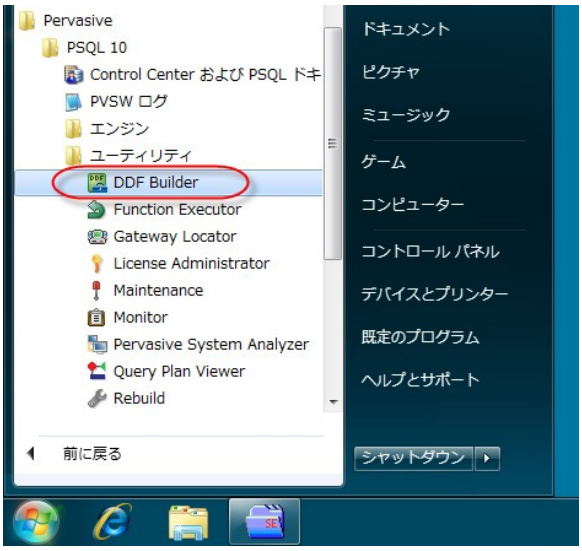
7.1. DDF Builder の起動

DDF Builder は、PCC から、あるいは Windows のスタートメニューから起動することができます。

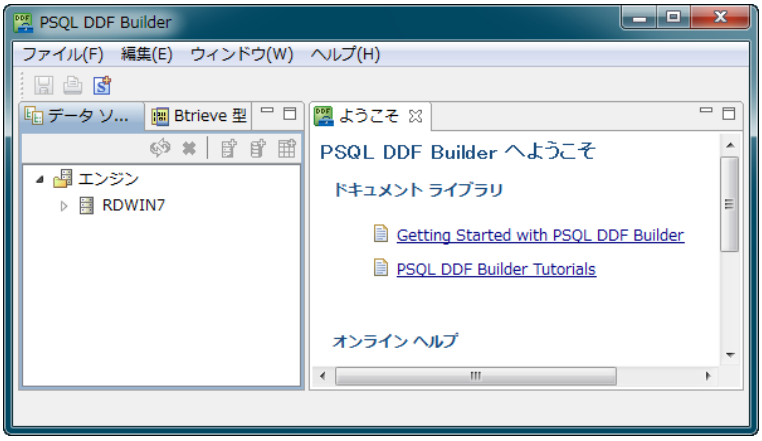
PCC の「ツール(T)」メニューから、「DDF Builder」を選択します。



あるいは、Windows のスタートメニューから、Pervasive → PSQL 10 → ユーティリティ → DDF Builder を選びます。



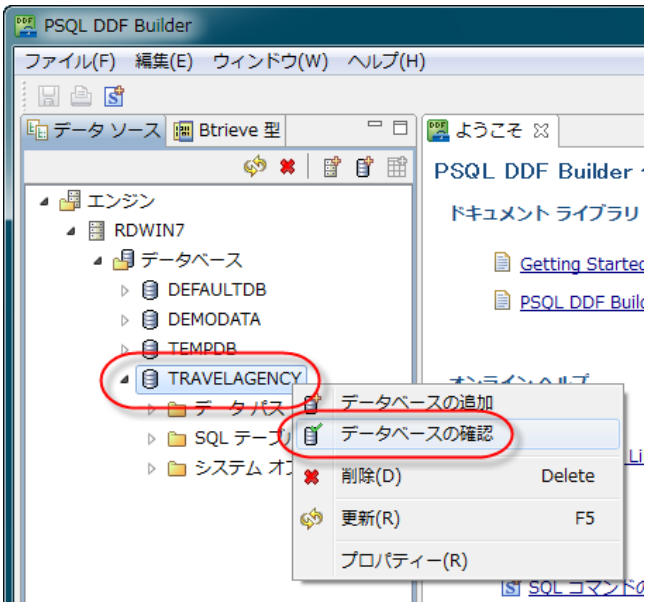
DDF Builder のウィンドウが開きます。



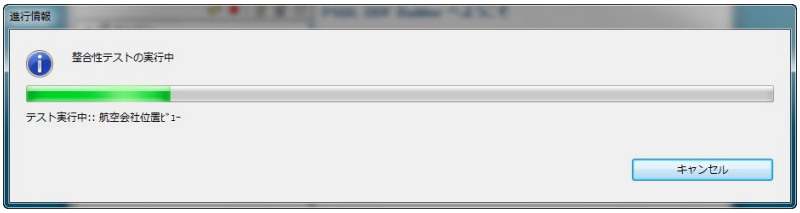
7.2. データベース全体の一括確認

最初に、作成したデータベース全体の整合性がとれているか、一括して確認しましょう。

DDF Builder 画面で、データベースを選び、ポップアップメニューから「データベースの確認」を選択します。

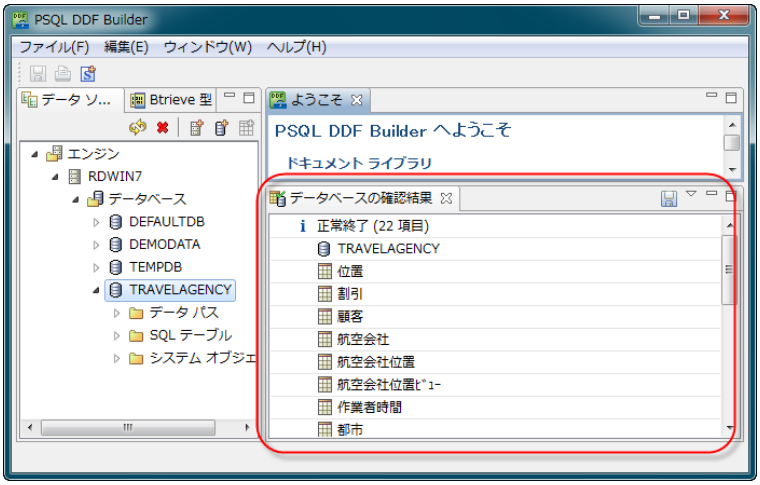


データベースやテーブルに対して、整合性テストが行われます。



結果は、「データベースの確認結果」欄に表示されます。
ここで、すべてが正常であることを確認してください。

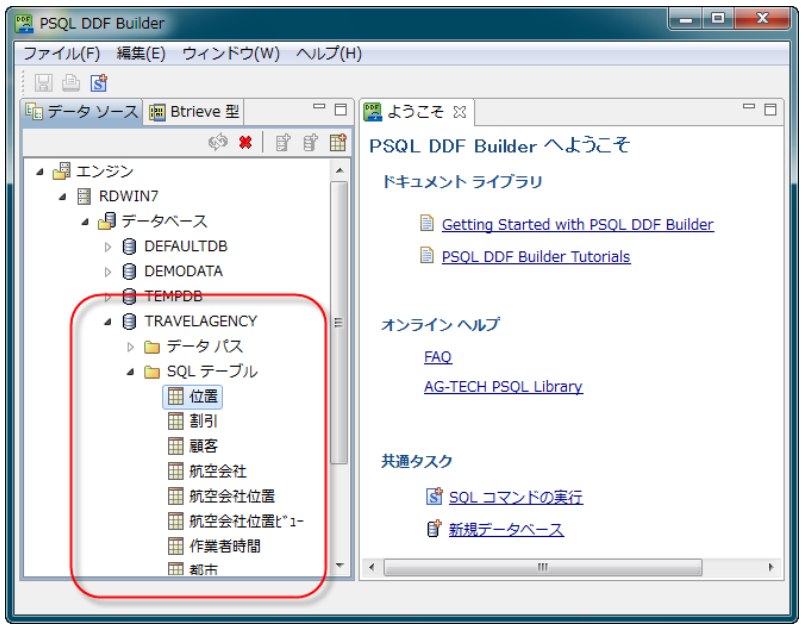
もし、エラーが見つかった場合には、次節の方法により、エラーの起きたテーブルについて、より詳細に確認します。



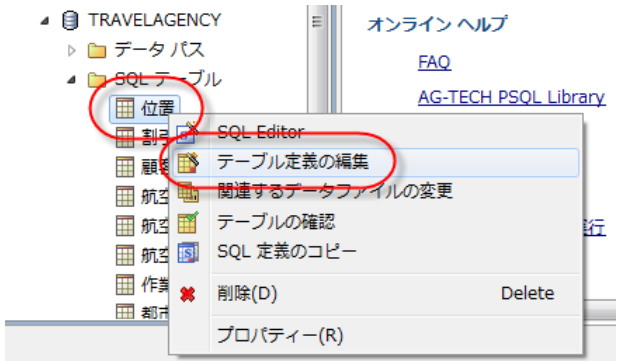
7.3. 個々のテーブルの確認

DDF Builder を使って、個々のテーブルについて、カラムやインデックス、データなどを詳細に確認することもできます。

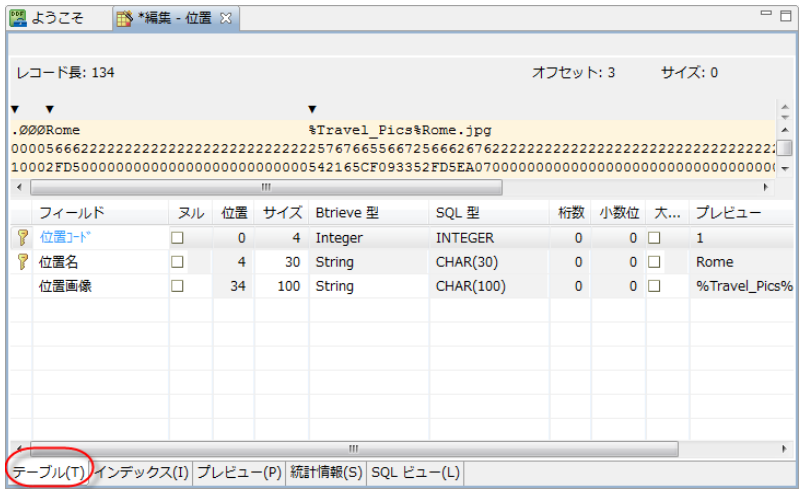
DDF Builder で、データベースを選び、「SQL テーブル」を開くと、定義されているテーブルの一覧が表示されます。



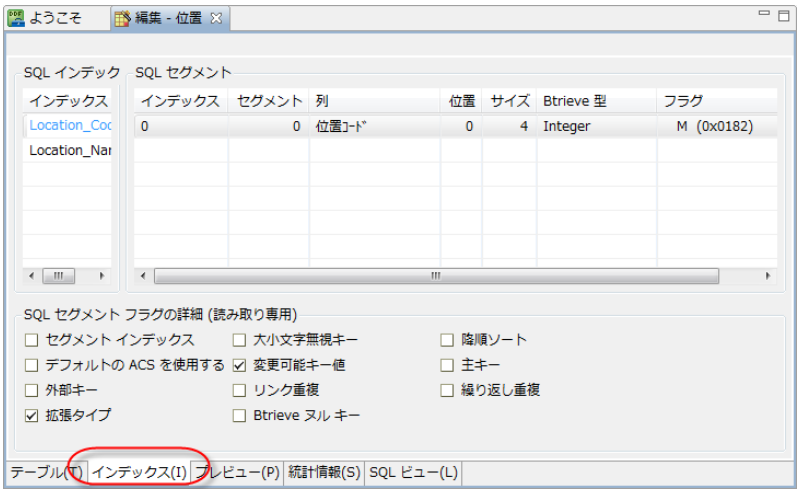
確認したいテーブルを選び、ポップアップメニューから「テーブル定義の編集」を選択します。



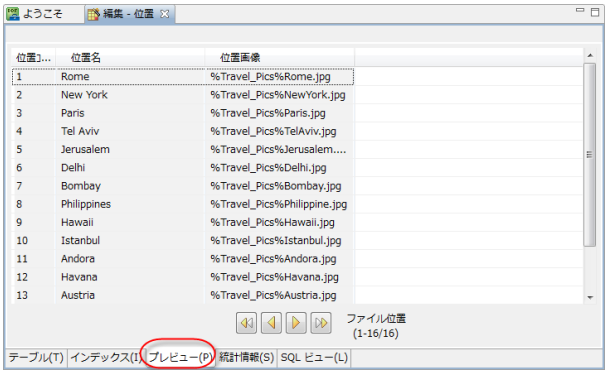
「テーブル」タブでは、テーブルのカラムの定義が表示されます。
データ型 (Btrieve 型、SQL 型の対応)、サイズを確認してください。



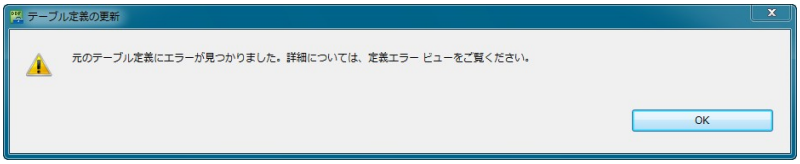
「インデックス」タブでは、インデックス定義が表示されます。各インデックスについて、セグメントのサイズとデータ型（Btrieve 型）が正しいか、確認してください。



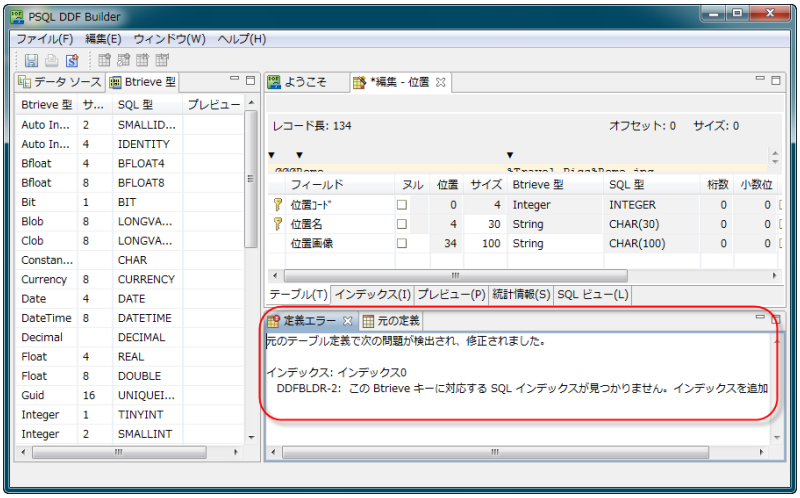
「プレビュー」タブでは、データの内容を表示します。
データが正しく表示されていることを確認してください。



もし、DDL によって定義されたメタデータと、実際の Btrieve ファイルの物理データ構造とに違いがある場合には、右図のようなエラーダイアログが表示されます。



エラーの内容は、「定義エラー」タブにも表示されます。



以上の確認がすべて OK ならば、作業終了です。

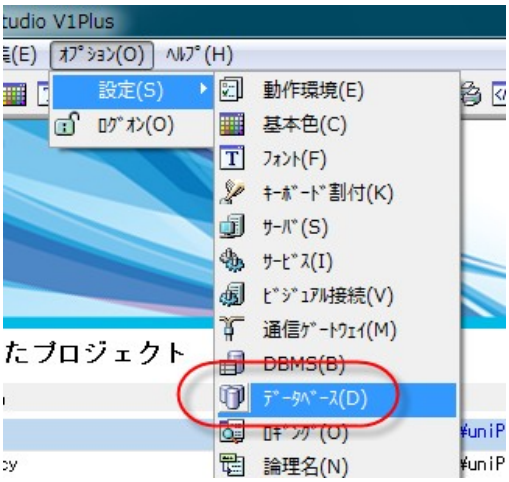
8. SQL テーブルを uniPaaS からアクセスする

SQL テーブルとして定義された Btrieve ファイルを、Pervasive SQL ゲートウェイを通して、uniPaaS からアクセスしてみましょう。

8.1. データベーステーブルの定義

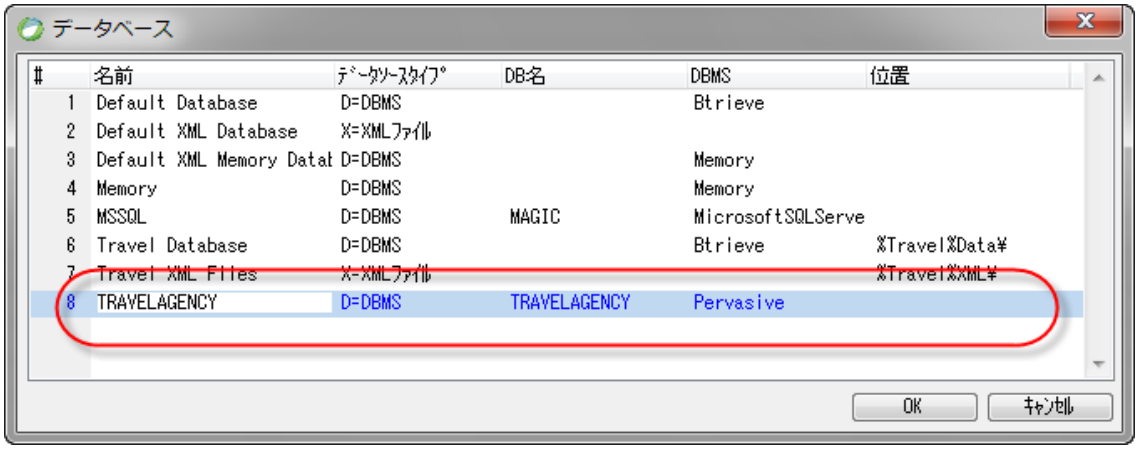
最初に、「データベース」テーブルにデータベースを定義します。

uniPaaS Studio を起動し、メニュー「オプション → 設定 → データベース」を選びます。
データベーステーブルが開きます。

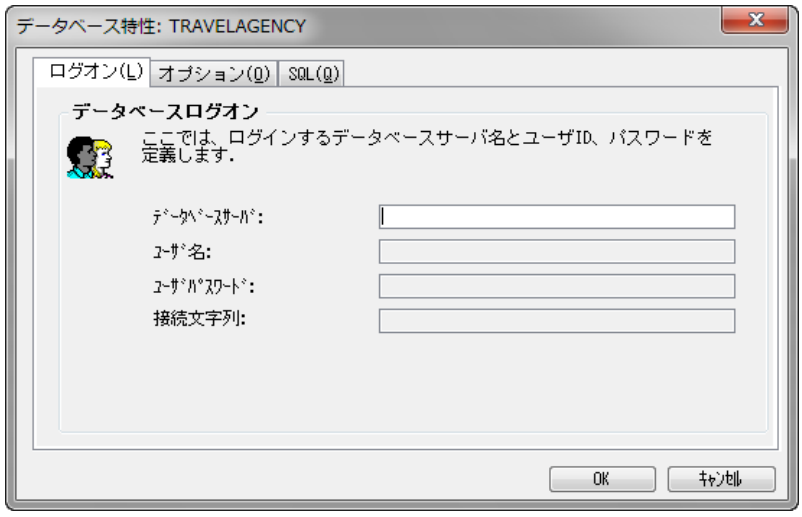


エントリを一つ作成して、右のように設定します。

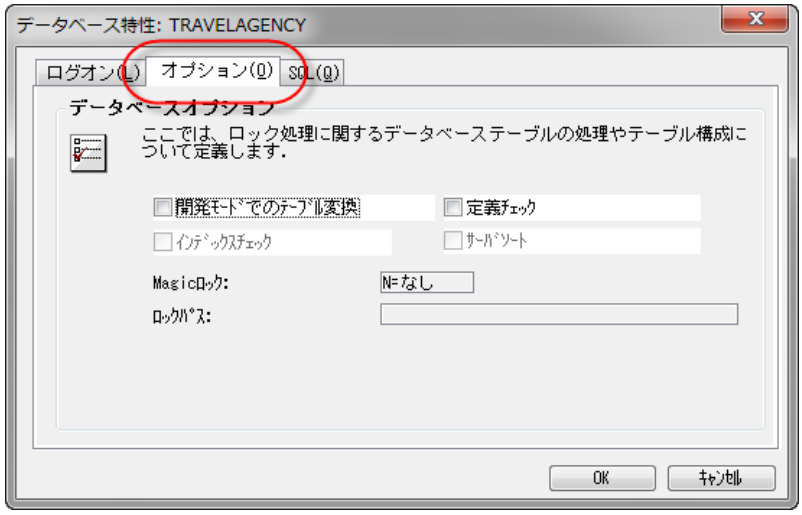
名前:	データリポジトリで参照する、任意の名前。
データソースタイプ:	D=DBMS を選びます。
DB 名:	ODBC に定義されたデータソース名 (DSN) を指定します。 ※ これは、PCC でデータソースを作成したときのデータベース名と同じです。
DBMS:	ズームして、「Pervasive」を選びます。
位置:	空欄のままにしておきます。



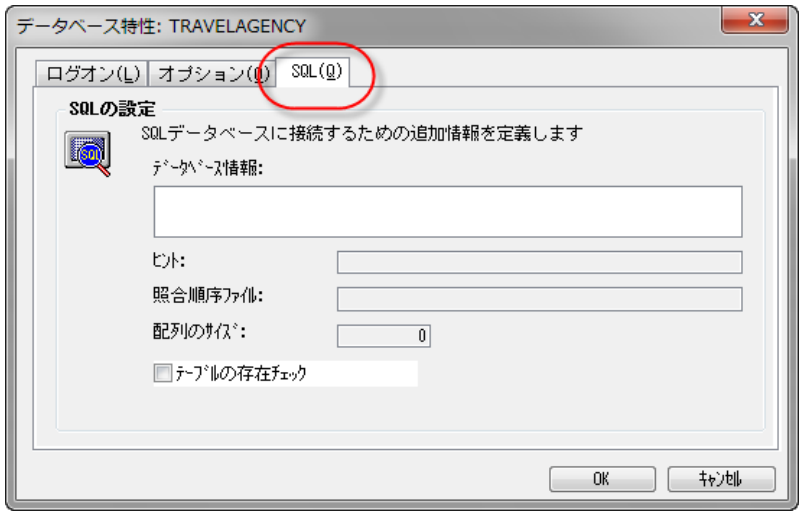
Alt+Enter を押して、データベース特性を開きます。
「ログオン」タブは、すべて空欄のままにしておきます。



「オプション」タブでは、すべてチェックを外しておいてください。



「SQL」タブでは、「テーブルの存在チェック」は、必要ないのでオフにします。
その他は空欄のままにしておきます。

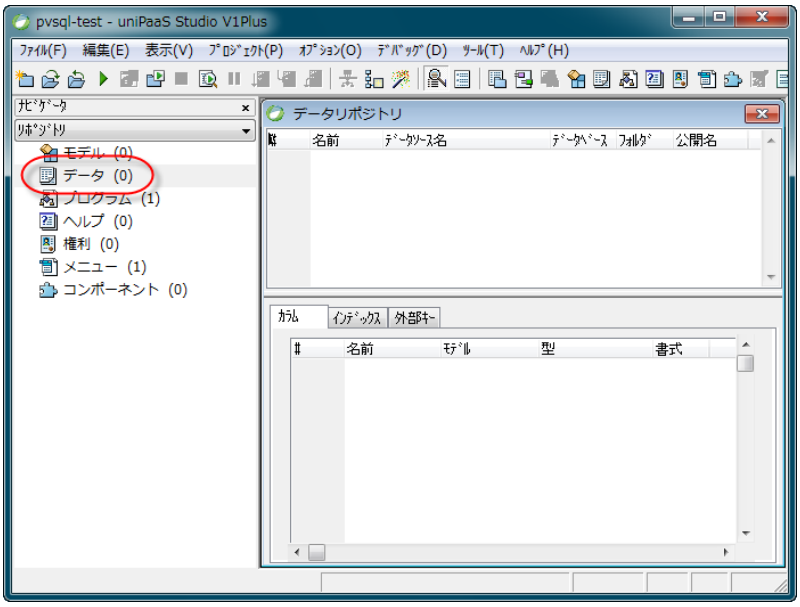


以上で、データベース テーブルの設定は終わりです。

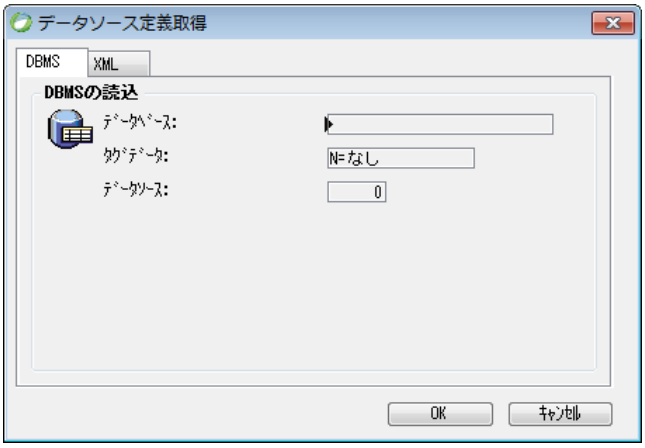
8.2. 定義取得

データベースから、テーブルの定義取得をします。

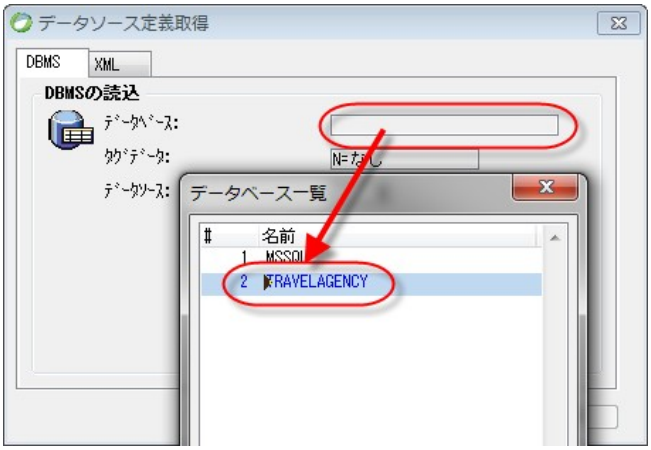
新しいプロジェクトを作成して、データリポジトリを開きます。



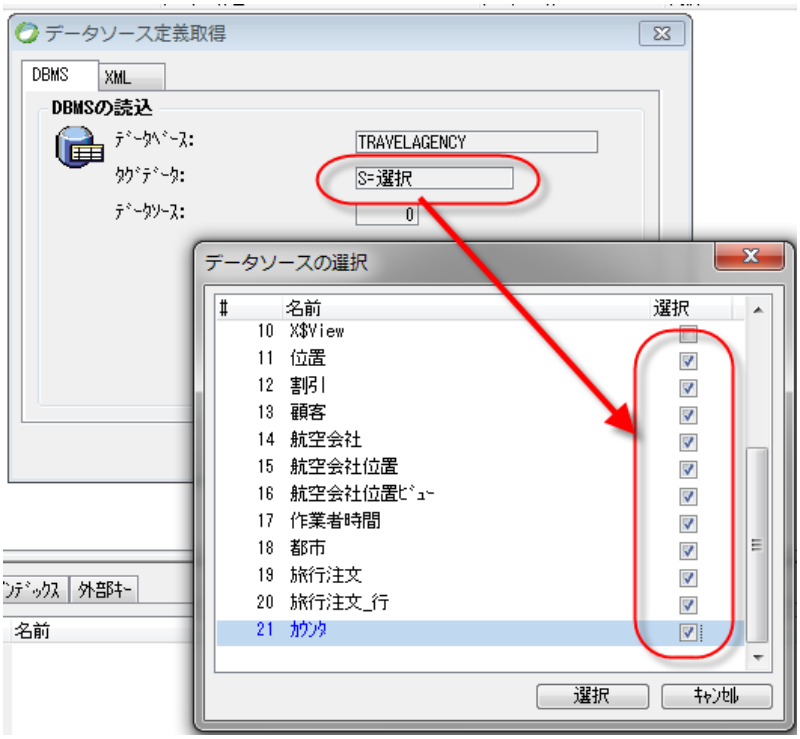
F9 キーを押して、定義取得ダイアログを開きます。



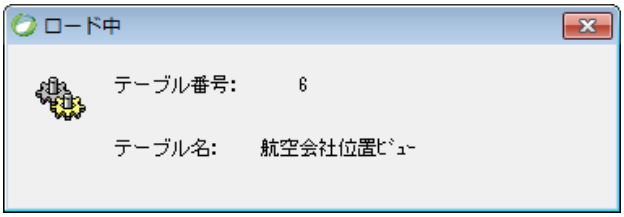
「データベース」欄でズームして、先に定義したデータベースを選択します。



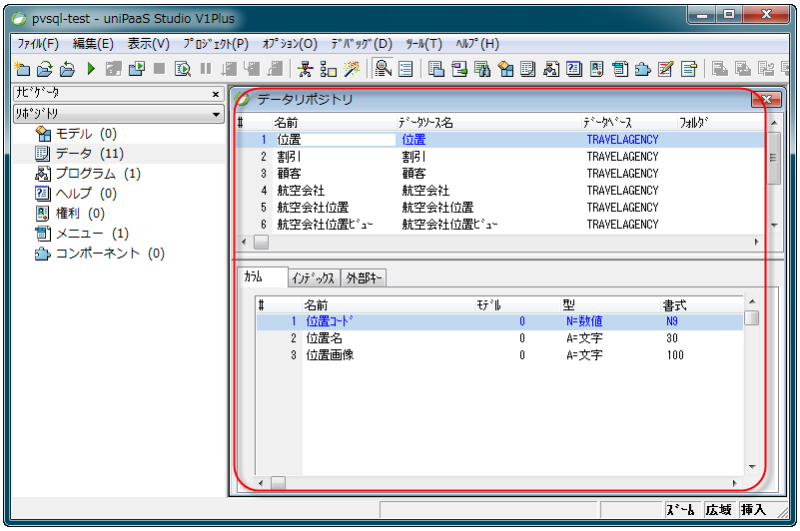
「タグデータ」欄で「S=選択」を選ぶと、テーブル一覧が表示されるので、定義取得したいテーブルを選択します。



データソース定義取得ダイアログに戻り、「OK」ボタンを押すと、定義取得が実行されます。



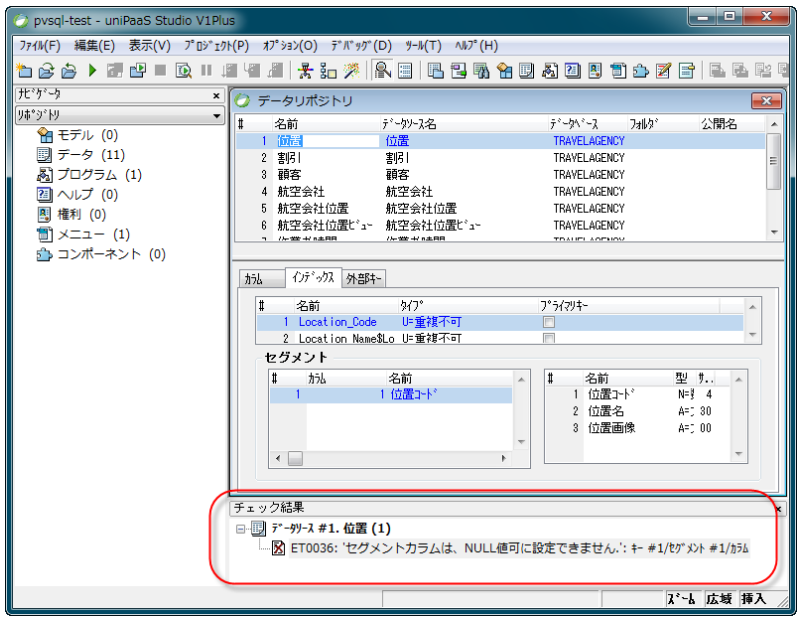
定義取得結果を確認してください。



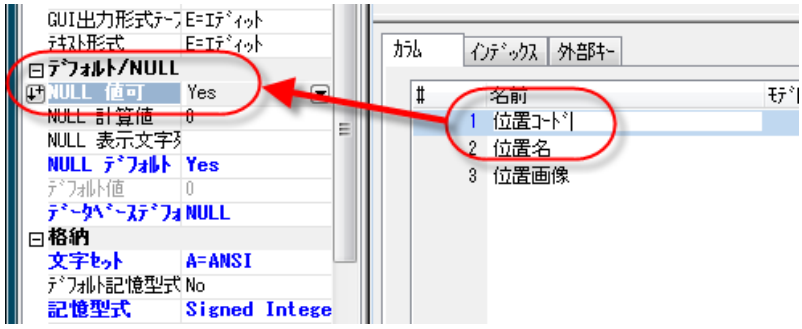
8.3. NULL 値可 特性の変更

ここで、F8 キーを押して、シンタックスチェックを行うと、「セグメントカラムは NULL 値可に設定できません」というエラーが出ます。

この例では、カラム 1「位置コード」が、インデックスのセグメントとして設定されているのに、「NULL 値可」＝ Yes として設定されているので、違反とされています。

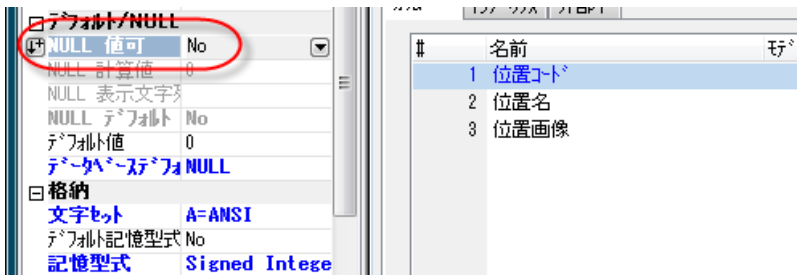


カラム 1「位置コード」の「NULL 値可」特性を見てみると、確かに、Yes になっています。



Pervasive.PSQL API の仕様として、Btrieve ファイルを SQL テーブルとしてアクセスするようにした場合、定義取得すると、NULL 値可 Yes という情報が取得されます。uniPaaS の Pervasive ゲートウェイは、この情報をもとにしてデータリポジトリのカラム特性を設定するので、このようになります。
しかし、実際には NULL 値は入れることができません。また、インデックスセグメントは、すべて NULL 値可 ＝ No のカラムでなければならい、という条件があるために、シンタックスチェックで引っかかるようになってしまいます。

この問題については、自動的に対応することはできませんので、インデックスカラムをひとつづつ手作業で「NULL 値可」＝No に設定してください。

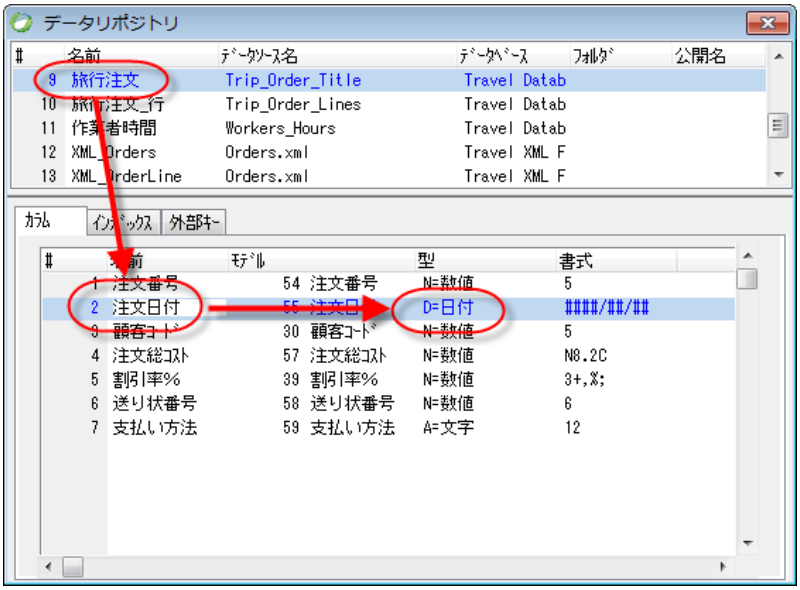


8.4. データ型の調整

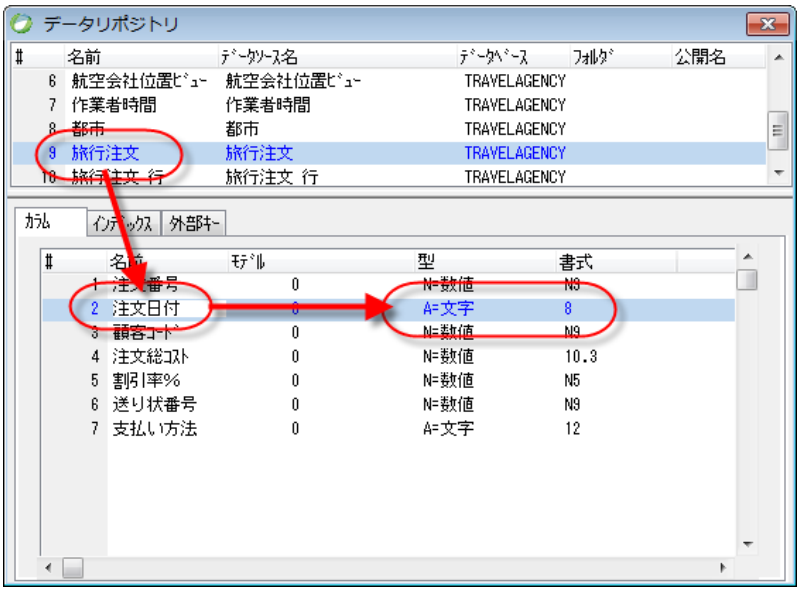
uniPaaS での Btrieve データの扱いと、Pervasive SQL でのデータの扱いとが異なる場合があるので、定義取得した直後には、もとの Btrieve とは異なるデータ型として登録されていることがあります。

8.4.1. データ型の異なる例とその理由

例えば、「旅行注文」テーブルの「注文日付」カラムを見てみると、もとの TravelAgency プロジェクトのデータリポジトリでは、「日付」型として定義されていました。



しかし、定義取得した後のテーブルを見てみると、書式「8」の文字型として登録されています。



これは、もとの Btrieve データで、日付データを「String Date」という格納形式で格納していたからです。String Date は、8バイトの文字型で、日付を YYYYMMDD の形式で格納するものです。
この Btrieve ファイルを SQL テーブルとして定義するときには、この日付カラムを CHAR(8) として定義します。DDLMaker で作成した DDL 文 (PVDDL.sql ファイル中に定義されています) を見てみると、次のようになっています。

```
/* ==== テーブル #9、テーブル名 "旅行注文" ==== */
SET TRUENULLCREATE = OFF;
```

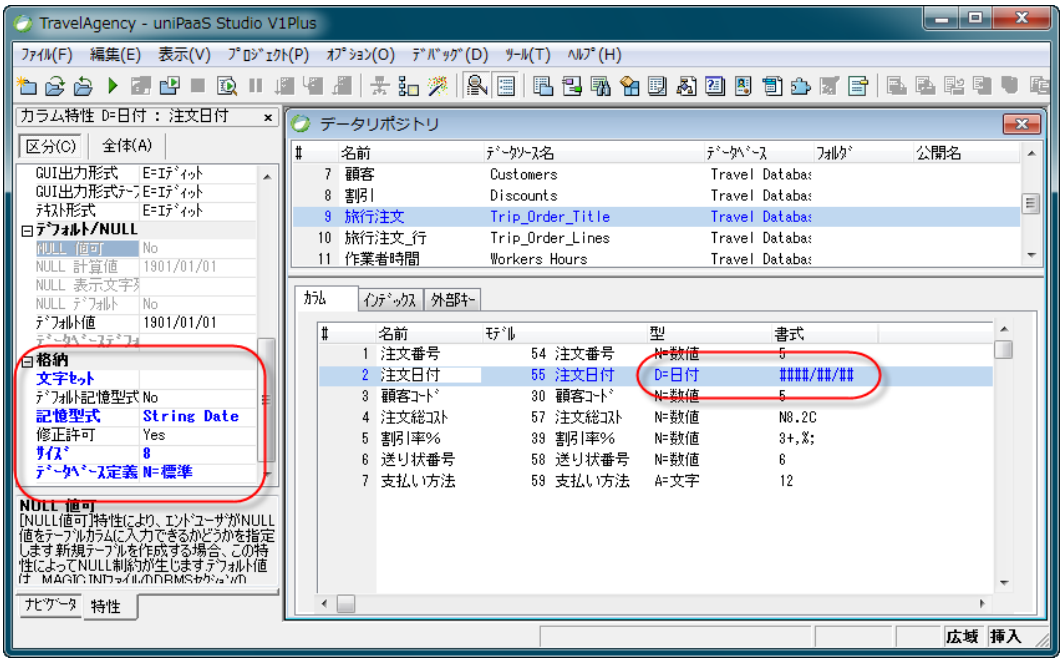
```
CREATE TABLE  "旅行注文"
IN DICTIONARY USING 'Trip_Order_Title'
(
   "注文番号"              INTEGER             /* 格納形式: NUM_SIGNED       書式:              サイズ:      4 */
  ,"注文日付"              CHAR(8)             /* 格納形式: DATE_STRING      書式:              サイズ:      8 */
   /*  格納形式 DATE_STRING を Pervasive の CHAR として定義します */
  ,"顧客ｺｰﾄﾞ"             INTEGER             /* 格納形式: NUM_SIGNED       書式: 5            サイズ:      4 */
  ,"注文総ｺｽﾄ"            DOUBLE              /* 格納形式: NUM_FLOAT        書式:              サイズ:      8 */
  ,"割引率%"               SMALLINT            /* 格納形式: NUM_SIGNED       書式: 3+,%;        サイズ:      2 */
  ,"送り状番号"            INTEGER             /* 格納形式: NUM_SIGNED       書式:              サイズ:      4 */
  ,"支払い方法"            CHAR(12)            /* 格納形式: ALPHA_STRING     書式:              サイズ:     12 */
);
SET TRUENULLCREATE = ON;
CREATE UNIQUE INDEX "Order_Number"
  IN DICTIONARY ON "旅行注文"
  (
     "注文番号"
  );
/* テーブル #9 終了。ステータス=S */
```

この赤色の文字の部分を見てわかるように、「注文日付」は CHAR(8) になっています。
その次の行に、コメントで「/* 格納形式 DATE_STRING を Pervasive の CHAR として定義します */」とあるのは、uniPaaS でのデータ型（日付型）と、Pervasive PSQL 上でのデータ型（CHAR 型）の間に、互換性がないことを警告しているものです。

このままでデータを表示させてみると、右図のように、日付のようなデータにはなっているものの、あくまで文字型であり、日付データとしてプログラムから利用することができません。

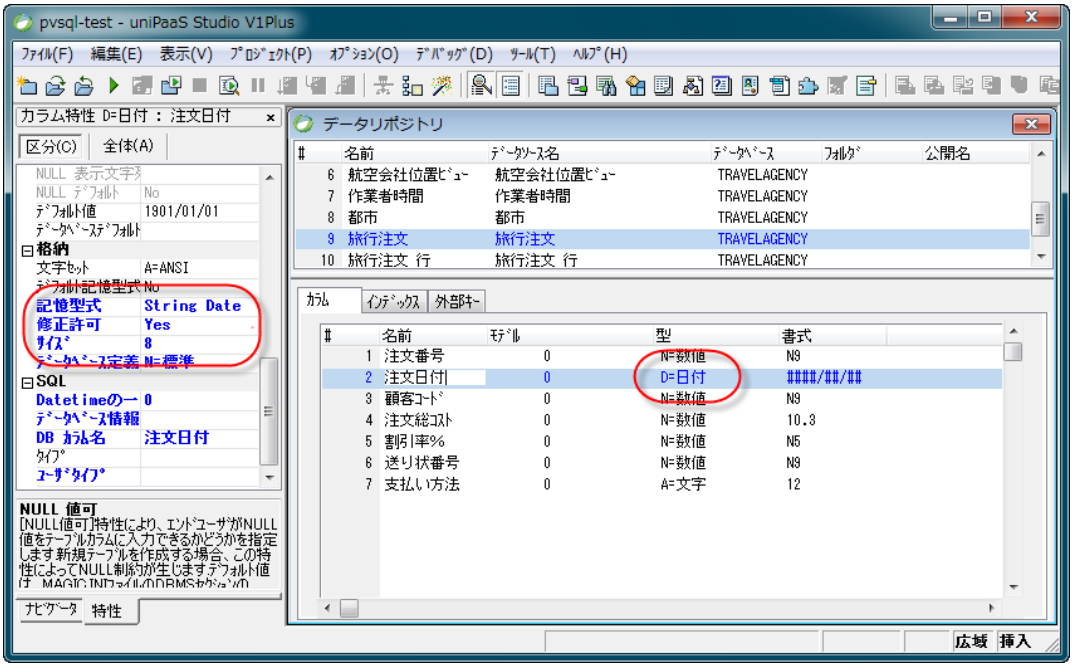
8.4.2. 対応方法

このような場合には、定義取得後のカラム定義を変更して Btrieve での定義に合わせるように調整する必要があります。
基本として、Btrieve での「型」、「書式」、「記憶型式」、「サイズ」と同じ設定をコピーします。
今の例では、Btrieve での定義は次のようになっていました。

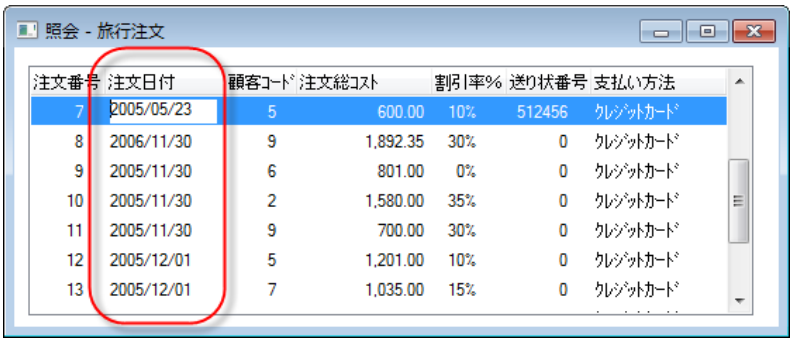


設定	値
型	D=日付
書式	####/##/##
記憶型式	String Date
サイズ	8

定義取得後のデータリポジトリでは、A=文字型になっているので、これを上記表に合わせて変更します。



こうして変更した後にデータを見てみると、書式通りに表示されています。
試しに、修正モードになって、不正な日付（5 月 32 日とか）を入力してみると、「不正な日付です」とエラーがでるので、正しく、日付データとして認識されていることがわかります。



この日付の例での調整は、データ型を変更するだけで、比較的簡単でしたが、項目の定義によっては、このように簡単でない場合もあります。そのような場合には、「記憶型式」についてのより深い理解が必要となります。
リファレンスマニュアル「ホーム > Magic uniPaaS リファレンス > Magic uniPaaS で使用するデータベース > Pervasive(SQL)データベース」の情報を基にして設定する必要となります。

9. DDLMaker の出力

4「DDLMaker の使い方」で説明したように、DDLMaker は、次の 2 つのテキストファイルを出力します。

- PVDDL.sql　　生成された DDL 文が格納されます。DDL 文は、そのまま PCC で実行できるような形式になっています。
- DDLMaker_(日付)(時刻).log　　処理の状態を記録するログが格納されます。エラーや警告などもこちらに含まれます。

9.1. PVDDL.sql の内容

PVDDL.sql ファイルには、次のような形式で CREATE TABLE/INDEX 文が格納されています。(左側の行数は説明用のもので、ファイル中にはありません)

```
1.  /* ==== テーブル #7、テーブル名 "顧客" ==== */
2.  SET TRUENULLCREATE = OFF;
3.  CREATE TABLE  "顧客"
4.  IN DICTIONARY USING 'Customers'
5.  (
6.     "顧客ｺｰﾄﾞ"              INTEGER          /* 格納形式: NUM_SIGNED       書式:              サイズ:     4 */
7.    ,"顧客名"                 CHAR(30)         /* 格納形式: ALPHA_STRING     書式:              サイズ:    30 */
8.    ,"電話番号"               CHAR(13)         /* 格納形式: ALPHA_STRING     書式:              サイズ:    13 */
9.    ,"住所"                   CHAR(30)         /* 格納形式: ALPHA_STRING     書式:              サイズ:    30 */
10.   ,"ｱﾊﾟｰﾄ名"               CHAR(3)          /* 格納形式: ALPHA_STRING     書式:              サイズ:     3 */
11.   ,"都市ｺｰﾄﾞ"              INTEGER          /* 格納形式: NUM_SIGNED       書式:              サイズ:     4 */
12.   ,"郵便番号"               INTEGER          /* 格納形式: NUM_SIGNED       書式:              サイズ:     4 */
13.   ,"誕生日"                 CHAR(8)          /* 格納形式: DATE_STRING      書式:              サイズ:     8 */
14.    /*  格納形式 DATE_STRING をPervasiveの CHAR として定義します */
15.   ,"顧客性別"               CHAR(6)          /* 格納形式: ALPHA_STRING     書式:              サイズ:     6 */
16.   ,"ｺﾞｰﾙﾄﾞ顧客"             LOGICAL          /* 格納形式: BOOLEAN_INTEGER  書式:              サイズ:     1 */
17.   ,"婚姻状況"               CHAR(8)          /* 格納形式: ALPHA_STRING     書式:              サイズ:     8 */
18.   ,"嗜好"                   CHAR(10)         /* 格納形式: ALPHA_STRING     書式:              サイズ:    10 */
19.   ,"好みの読書"             CHAR(10)         /* 格納形式: ALPHA_STRING     書式:              サイズ:    10 */
20.   ,"好みの音楽"             CHAR(10)         /* 格納形式: ALPHA_STRING     書式:              サイズ:    10 */
21. );
22. SET TRUENULLCREATE = ON;
23. CREATE UNIQUE INDEX "Customer_Code"
24.   IN DICTIONARY ON "顧客"
25.   (
26.     "顧客ｺｰﾄﾞ"
27.   );
28. /* テーブル #7終了。ステータス=S */
29.
30. CREATE UNIQUE INDEX "Customer_Name$Custom"
31.   IN DICTIONARY ON "顧客"
32.   (
33.     "顧客名"
34.    ,"顧客ｺｰﾄﾞ"
35.   );
36. /* テーブル #7終了。ステータス=S */
```

- 1 行目はコメントで、uniPaaS データリポジトリ中でのテーブルの番号、テーブル名が記録されます。
- 2 行目と 22 行目の SET TRUENULLCREATE 文は、Btrieve ファイルの定義のために必要なものです。
- 3～21 行目が、CREATE TABLE 文です。
  - 「IN DICTIONARY 」句は、既存の Btrieve ファイルのメタデータの定義であることを示すために附加されています。

    - 「USING …」句は、Btrieve ファイルの OS ファイル名を示します。
- 6～20 行目には、各カラムの名前、データ型が定義されています。各行のコメントには、参考情報として、uniPaaS データリポジトリで定義されている格納形式、書式、およびサイズが記録されています。
- 14 行目には、データ型の互換性に関するコメントが記録されています。
  この例では、「誕生日」という名前のカラムが、uniPaaS では日付型として定義されていますが、格納形式が String Date なので、Pervasive PSQL では文字型 (CHAR(8))として定義されるので、このカラムを扱う場合には、データの互換性に注意が必要なことを示しています。
- 23～27 行目は、CREATE INDEX 文で、インデックスの構造を定義するものです。ここにも、既存の Btrieve ファイルのメタデータ定義であることを示すために、「IN DICTIONARY」句が付いています。
- 26 行目には、インデックスのセグメントを構成するカラム名が列挙されています。
- 28 行目は、ひとつのテーブルの定義の終りを示すコメントです。

CREATE TABLE 文、および CREATE INDEX については、Pervasive PSQL のヘルプで、「上級者用ドキュメント > SQL Engine Reference > SQL 構文リファレンス」 の各項を参照してください。

9.2. ログファイル

DDLMaker_(日付)(時刻).log という名前のファイルは、DDLMaker が処理中に作成したログファイルです。ここにはエラーや警告などが記録されるので、必ず確認してください。

ログファイルは、CREATE TABLE/INDEX 文に似せた形式で記録されますが、実際の CREATE 文としては不完全なものなので、PCC で実行させようとしても正しく実行できません。

例1: 次は、TravelAgency のデータを変換したログファイルの一部です。
(文字の色分けは、説明のために入れたもので、ログファイル中にはありません)

```
1.  /* Pervasive.SQL用の CREATE TABLE/INDEX文を生成します。    */
2.  /* データソースファイル名：C:¥Program Files¥uniPaaS¥Studio V1Plus¥Projects¥TravelAgency¥Source¥DataS
3.
4.  /* ===== テーブル#      1 テーブル名： ｶｳﾝﾀ                              ===== */
5.  /* テーブル# 1       正常終了
6.
7.  (途中省略)
8.
9.  /* ===== テーブル#      7 テーブル名： 顧客                              ===== */
10. /* テーブル# 7       BINARYとして定義された項目があります
11.
12. (途中省略)
13.
14. /* DDL作成が終了しました。      */
```

2 行目は、uniPaaS のプロジェクトのデータリポジトリに対応する XML ファイル名を記録しています。
4～5 行目は、正常終了したテーブルの記録です。
9～10 行目は、DDL 文は正常に作成されたものの、Btrieve データに対応する Pervasive PSQL データ型がないために、一部のカラムを BINARY として定義したことを示しています。このような場合、DDL 文を PCC で実行して、Btrieve ファイルを SQL テーブルとしてアクセスすることは可能ですが、BINARY データはアプリケーションで加工する必要があります。

Btrieve テーブルの構造によっては、たとえ BINARY データ型を使ったとしても、Pervasive PSQL のテーブルとして変換が不可能なものもあります。そのような場合には、ログにそのエラー内容が記録され、PVDDL.sql ファイルには、DDL 文が記録されません。

例2: 以下は、Travel Agency でない、別のプロジェクトのデータリポジトリを使った場合のログの一部です。
この例では、書式 8001 の文字型カラムが変換できなかったことを表しています。Pervasive PSQL では、CHAR 型の最大長が 8000 なので、8001 バイトの Btrieve 文字型フィールドは変換が不可能です。
このようなカラムがある場合、無理に DDL 文を作って実行しようとすると、CREATE TABLE 文実行時にエラーとなるか、あるいは、DDL の定義と実際の Btrieve ファイルの物理構造とが相違して、データ破損の原因となりますので、DDLMaker では DDL 文を作成しません。

```
1.  /* ===== テーブル#      2 テーブル名： ALPHA8001                         ===== */
2.     /* ==== テーブル #2、テーブル名 "ALPHA8001" ==== */
3.     SET TRUENULLCREATE = OFF;
4.     CREATE TABLE  "ALPHA8001"
5.     IN DICTIONARY USING 'ALPHA8001'
6.     (
7.       "N"          INTEGER              /* 格納形式: NUM_SIGNED      書式: N5        サイズ:     4 */
8.      ,"A8001"      /*対応データ型なし*/ /* 格納形式: ALPHA_STRING    書式: 8001      サイズ:  8001 */
9.       /*  *** 長さが8000バイトを越えているので変換できません *** */
```

```
10.       ,"A20"          CHAR(20)              /* 格納形式: ALPHA_STRING      書式: 20          サイズ:     20 */
11.     );
12.     SET TRUENULLCREATE = ON;
13.     CREATE UNIQUE INDEX "A8000_BY_N"
14.       IN DICTIONARY ON "ALPHA8001"
15.       (
16.          "N"
17.       );
18.     /* テーブル #2終了。ステータス=C */
19.  /* テーブル# 2       対応するデータ型がないので変換できませんでした
```

例3: 次の例は、LString の格納形式の文字型カラムを変換しようとした時のエラー例です。Pervasive PSQL には、LString データ型はあるものの、最大長が 254 なので、255 バイトの変換ができなかったことを示しています。

```
1.   /* ===== テーブル#      4 テーブル名 :  ALPHALSTR255                    ===== */
2.      /* ==== テーブル #4、テーブル名 "ALPHALSTR255" ==== */
3.      SET TRUENULLCREATE = OFF;
4.      CREATE TABLE  "ALPHALSTR255"
5.      IN DICTIONARY USING 'ALPHLSTRA255'
6.      (
7.         "N"          INTEGER               /* 格納形式: NUM_SIGNED        書式: N5          サイズ:      4 */
8.        ,"A_L255"     /*対応データ型なし*/ /* 格納形式: ALPHA_LSTRING     書式: 255         サイズ:    255 */
9.         /*  *** エラー：対応データ型がないので正しく変換できません *** */
10.       ,"A_20"       CHAR(20)              /* 格納形式: ALPHA_STRING      書式: 20          サイズ:     20 */
11.     );
12.     SET TRUENULLCREATE = ON;
13.     CREATE UNIQUE INDEX "A8000_BY_N"
14.       IN DICTIONARY ON "ALPHALSTR255"
15.       (
16.          "N"
17.       );
18.     /* テーブル #4終了。ステータス=C */
19.  /* テーブル# 4       対応するデータ型がないので変換できませんでした
```

Magic uniPaaS V1Plus DDLMaker 利用ガイド

第 1 版　2011 年 8 月 24 日

発行　〒151-0053 東京都渋谷区代々木三丁目二十五番三号
あいおいニッセイ同和損保新宿ビル 14 階
マジック ソフトウェア・ジャパン (株)
http://www.magicsoftware.co.jp/