

はじめに

Caché を従来の MUMPS 処理系と比較した時の大きなメリットは、他の MUMPS 上のグローバル等を Caché に移行した後、そのグローバルに対して SQL で参照したり、オブジェクトアクセスを行うことができる点です。但し、その恩恵を得るためには、若干の設定（ストレージマッピング）が必要です。ここでは、2 種類の設定方法を紹介します。1 つは、スタジオに付属するストレージエディタを利用する方法、もうひとつは、ストレージマッピング定義を直接 XML 表現で記述する方法です。前者は、GUI による直感的な操作でわかりやすいですが、大量のマッピングを定義する際には、GUI 操作を繰り返す必要があり、煩雑になります。一方、後者は、構造を鳥瞰的に理解するのは難しく、記述ミスも起こりやすいですが、同様の定義を連続して行なう場合には、効率良く定義を行なうことが可能です。最初の定義は、前者の方法で行い、あとの繰り返しは、後者の方法を組み合わせるという方法が考えられます。

1. Caché スタジオストレージエディタを利用したマッピング定義方法

単純な構造のマッピング

以下のようなグローバルをマッピングします。

^KION(地域,日付)="朝の気温^昼の気温^夜の気温"

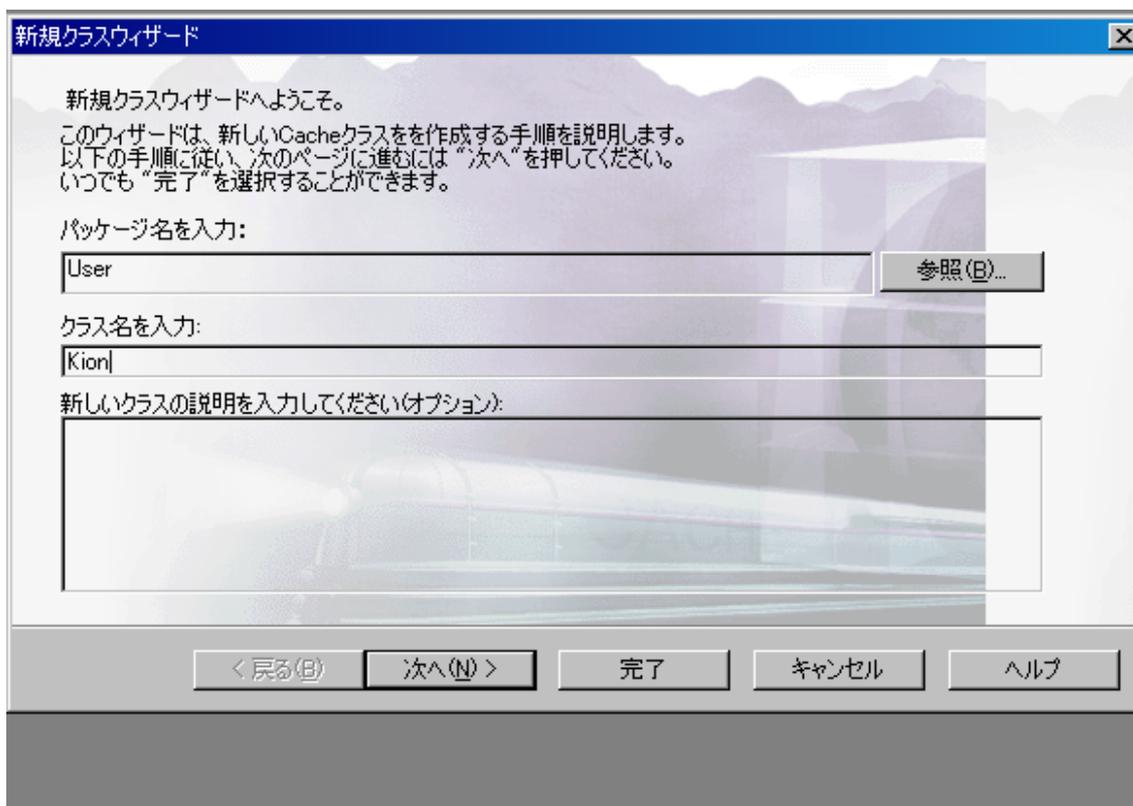
ex) ^KION("大阪",59311)="19^25^21"

^KION("東京",59312)="16^24^20"

クラス定義

まず普通に Caché スタジオでクラス定義します。

クラス名(User.Kion)



プロパティの定義

次に必要なプロパティを定義します。

プロパティ名をそれぞれ

Area,Kdate,Morning,Afternoon,Night (地域、日付、朝の気温、昼の気温、夜の気温)

とします。

```
Property Area As %String [ SqlFieldName = 地域 ];  
Property Kdate As %Date [ SqlFieldName = 日付 ];  
Property Morning As %Integer [ SqlFieldName = 朝の気温 ];  
Property Afternoon As %Integer [ SqlFieldName = 昼の気温 ];  
Property Night As %Integer [ SqlFieldName = 夜の気温 ];
```

新規プロパティウィザードへようこそ。
このウィザードはクラス定義に新しいプロパティを追加するお手伝いをします。以下の指示に従って、“次”を押すと次のページに移動します。いつでも“完了”を押すことができます。

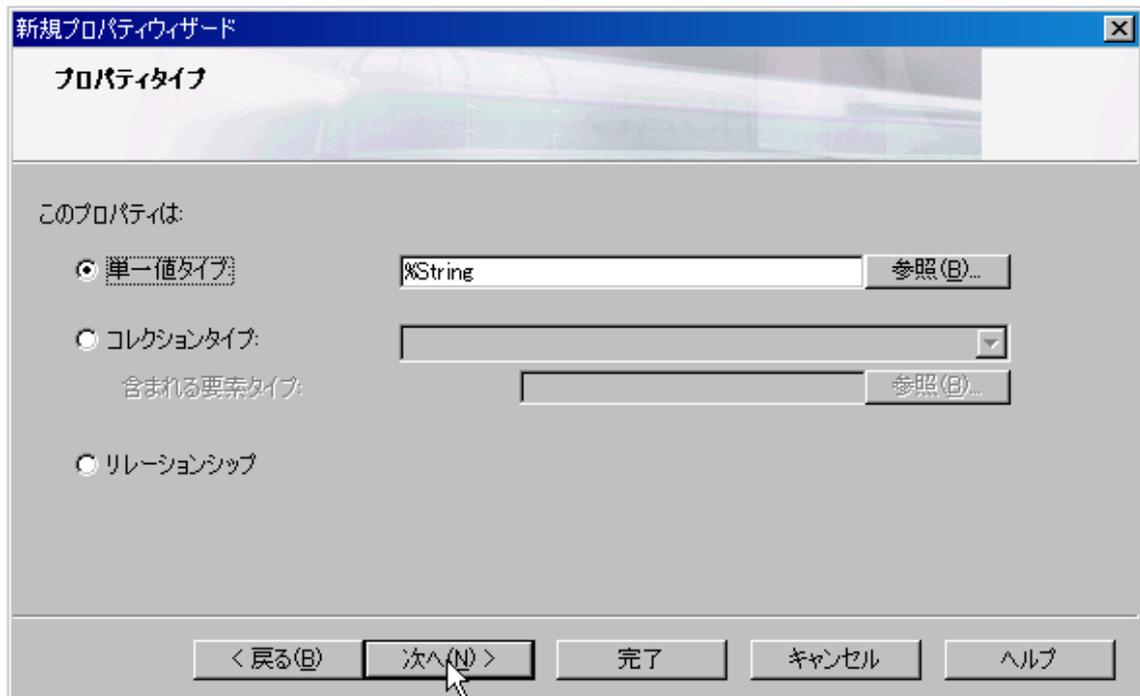
この新しいプロパティの名前を入力して下さい:

Area

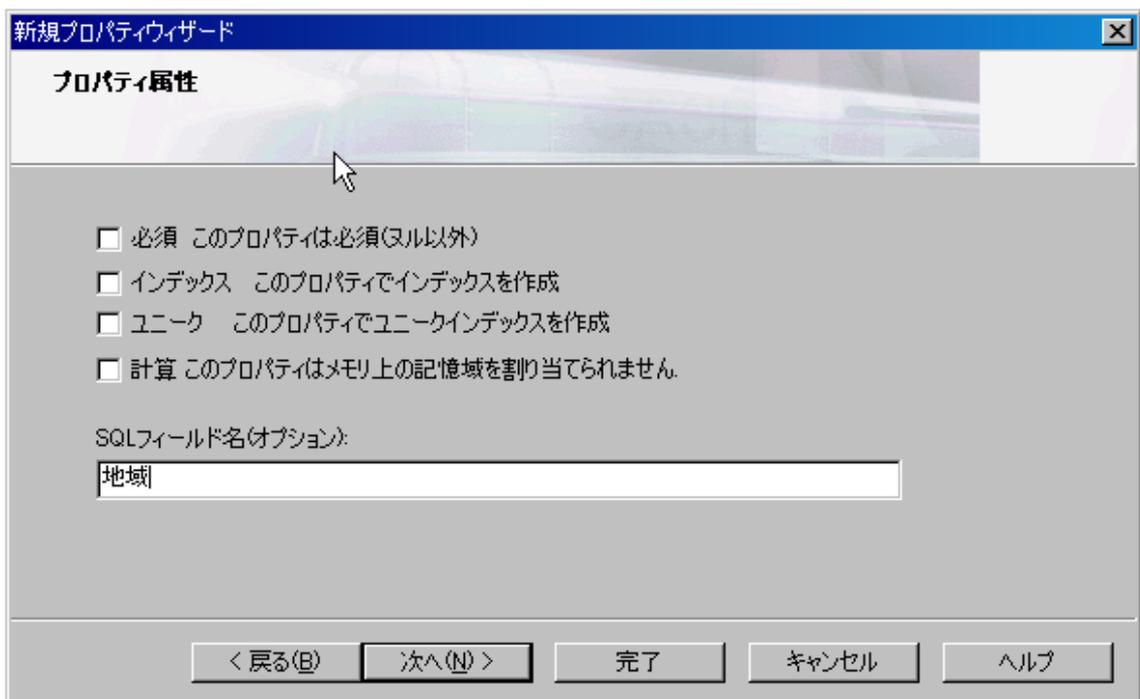
この新しいプロパティの説明を入力して下さい(任意):

< 戻る(B) 次へ(N) > 完了 キャンセル ヘルプ

各プロパティのデータタイプを選択します。



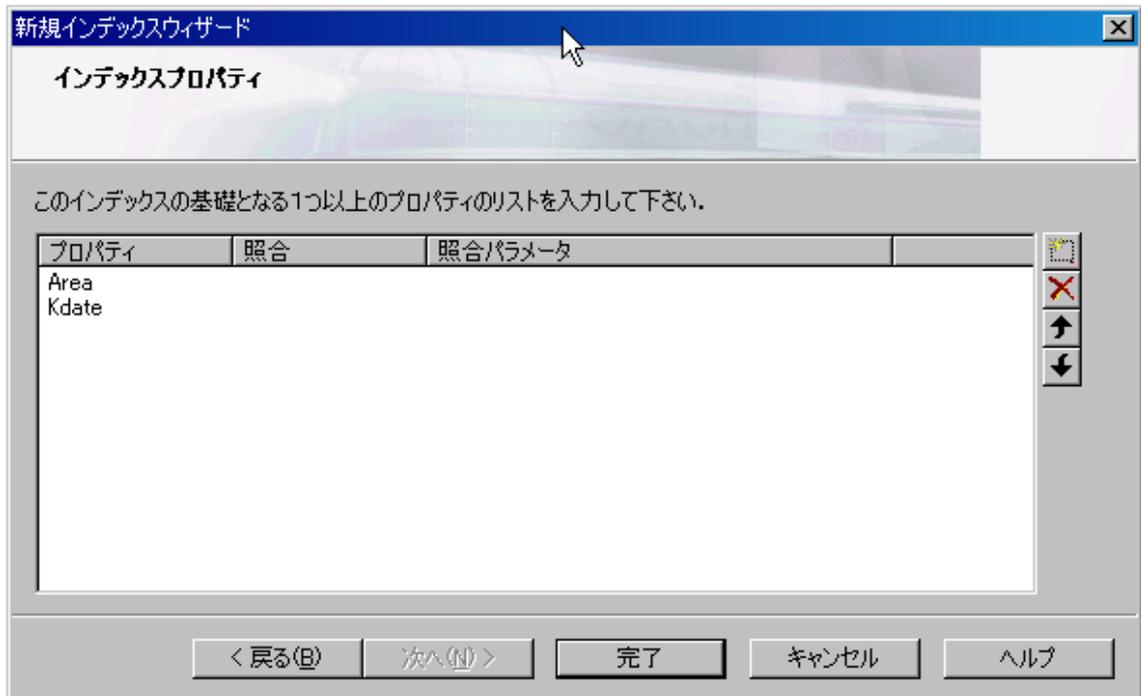
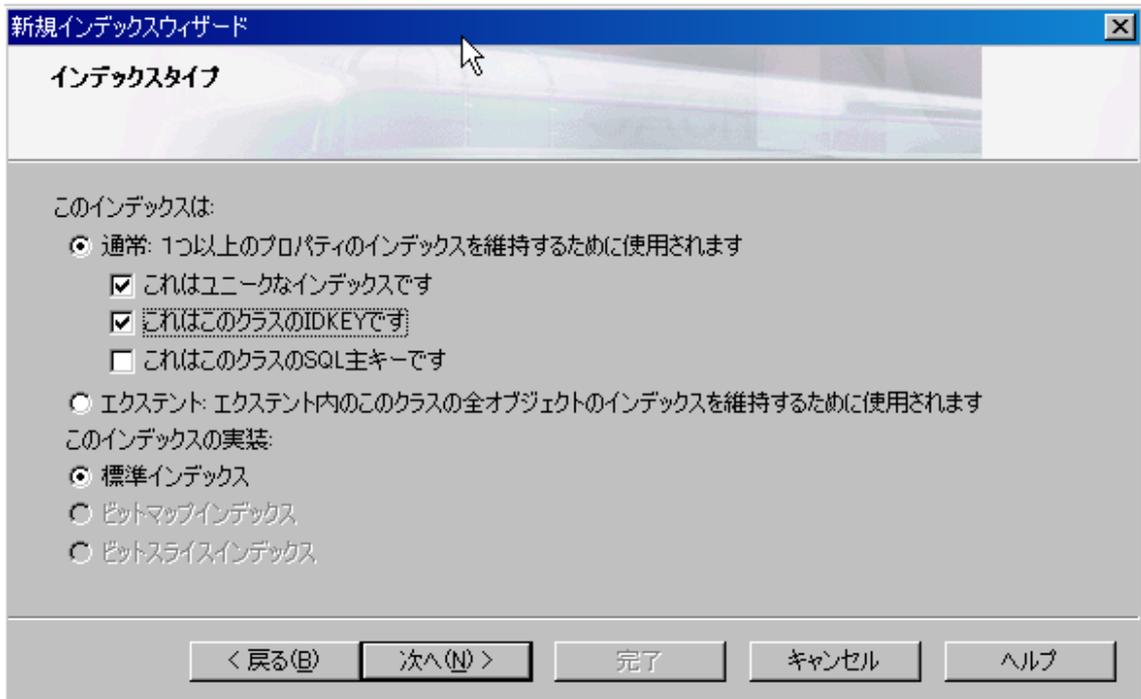
次に SQL カラム名に日本語を付けるために SQL フィールド名に入力します、



インデックスの定義

地域+日付データでデータが一意に決定するように、インデックス定義が必要です。

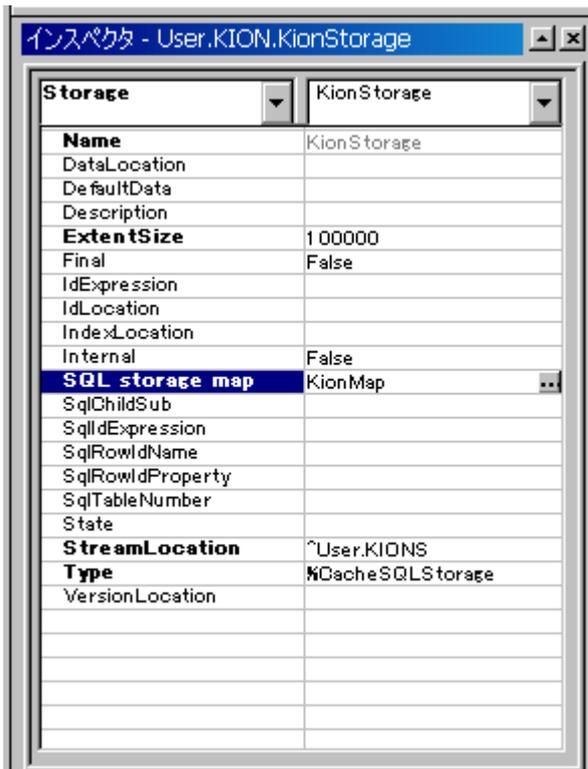
```
Index KionIndex On (Area, Kdate) [ IdKey, Unique ];
```



Storage の定義

次に Storage を設定します。メニューのクラス→追加→新規ストレージから新規ストレージウィザードを起動してください。

任意の名前を入力し、CacheSQL ストレージを選択してください。



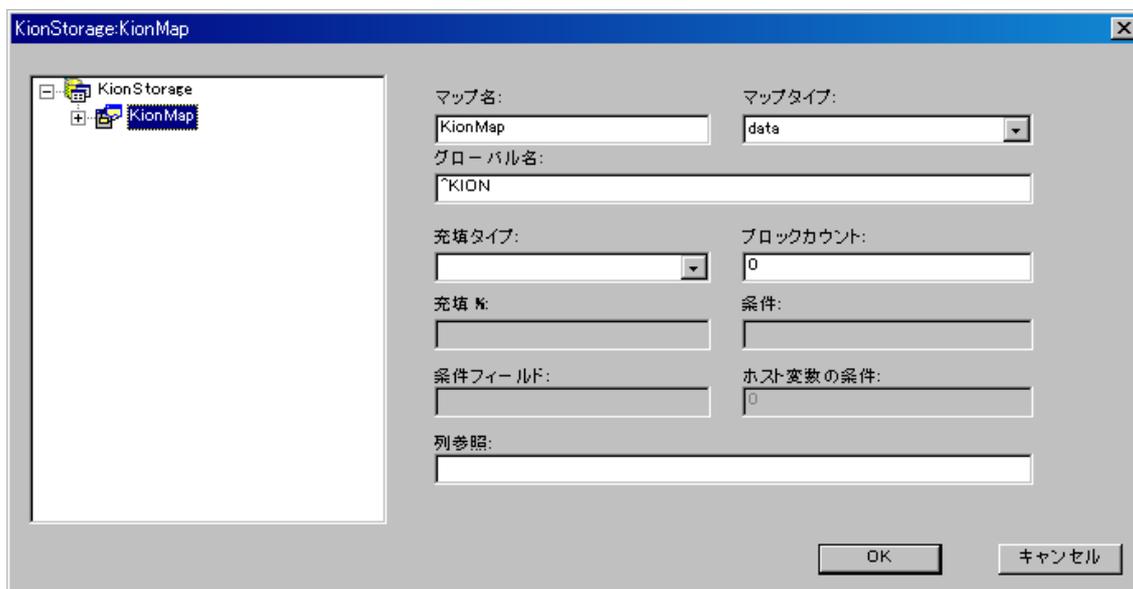
SQL storage 設定ウィンドウが起動します。



次にストレージマッピングを定義していきます。追加ボタンを押して基本情報（マップ名、マップタイプ、グローバル名、ノード構造）を定義します。

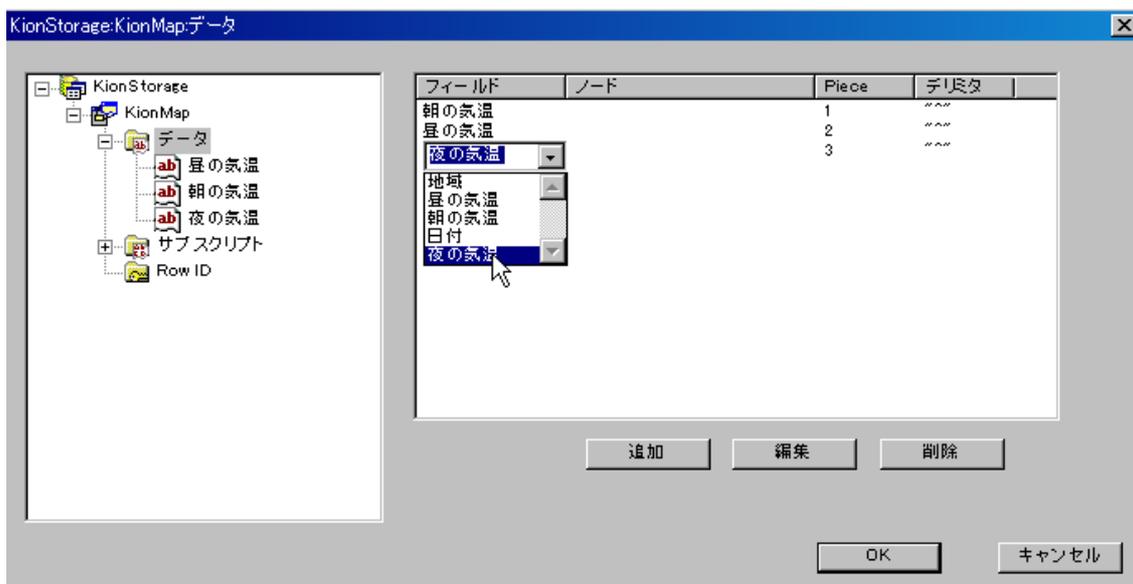
マップ名：任意 マップタイプ：data

グローバル名：^KION

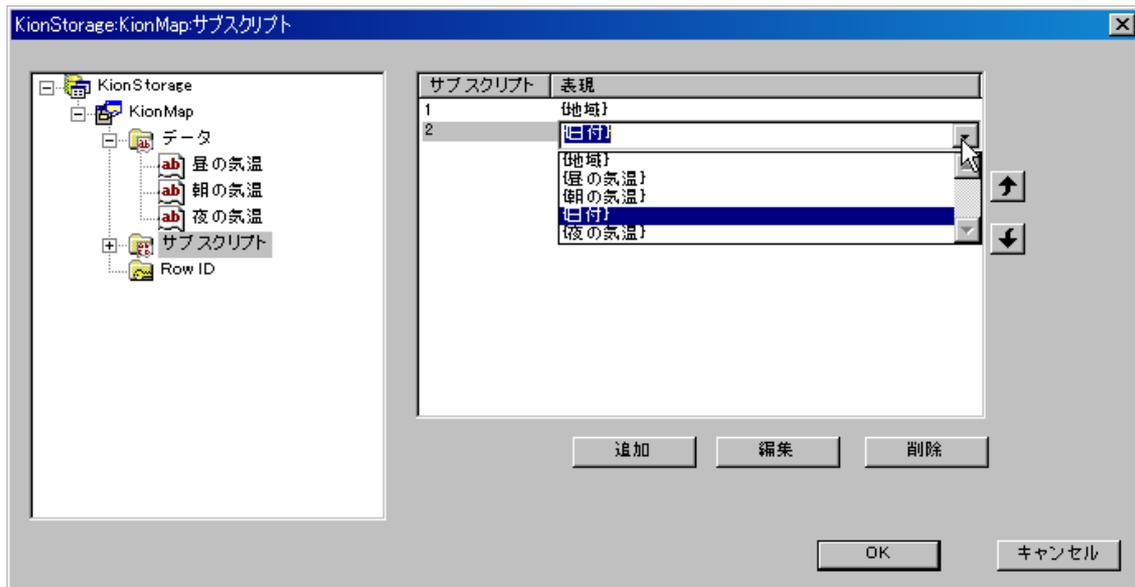


次に各項目（データ，サブスクリプト，RowID,）を定義していきます。

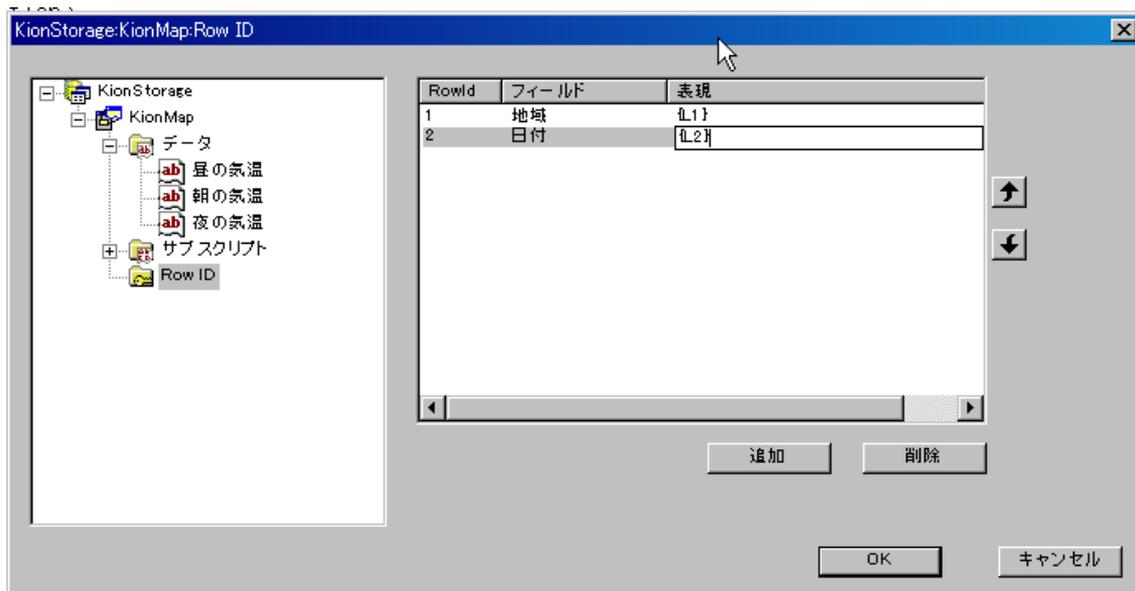
・データ



・サブスクリプト



• RowID



コンパイルを実行します。

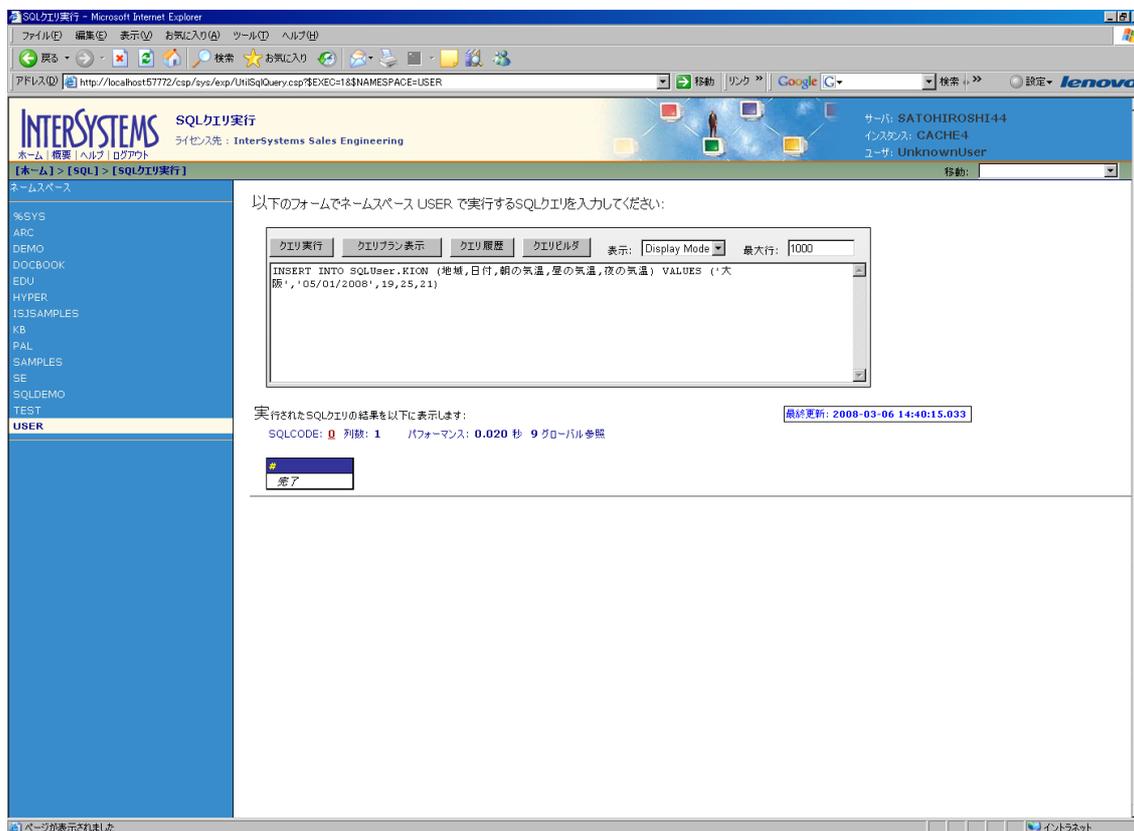
データの確認

- システム管理ポータルから参照 (データ管理->SQL)

データ管理から SQL を選択し、SQL 操作の画面で、左側からネームスペースを選択。

SQL 操作の SQL 文の実行を選択

まずテストデータの入力を行うために、SQL 文でデータをインサートします。



SQL 文の入力

INSERT INTO SQLUser.KION (地域,日付,朝の気温,昼の気温,夜の気温) VALUES ('大阪','05/01/2008',19,25,21)

表示： を Display Mode にします。

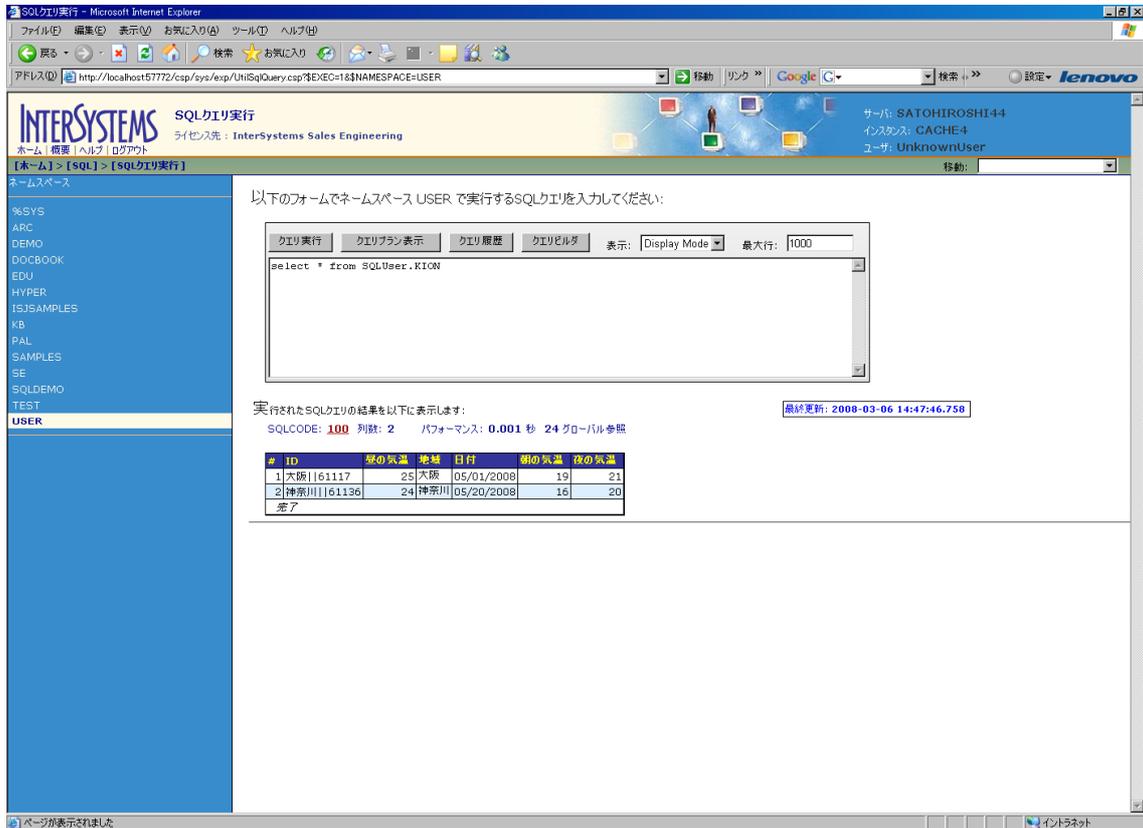
クエリの実行ボタンを押します。

同様に以下の SQL 文を入力して、クエリの実行ボタンを押します。

INSERT INTO SQLUser.KION (地域,日付,朝の気温,昼の気温,夜の気温) VALUES ('神奈川','05/20/2008',16,24,20)

次に SQL 文で内容が登録されているか確認します。

Select * from SQLUser.KION



- Cache ObjectScript から追加

今度は、テストデータをオブジェクトのインタフェースを使って、登録して、SQL 文で見ることができることを確認してみます。

```
USER>S kion=##class(User.KION).%New()
```

```
USER>s kion.Area="京都",kion.Kdate=$P($H,",",1),kion.Morning=20,kion.Afternoon=27,kion.Night=18
```

```
USER>s sts= kion.%Save()
```

- ・ システム管理ポータルから再度参照

SQLクエリ実行 - Microsoft Internet Explorer

アドレス http://localhost:57772/csp/sys/exp/UhlSqlQuery.csp?EXEC=1&NAMESPACE=USER

INTERSYSTEMS SQLクエリ実行
ライセンス先: InterSystems Sales Engineering

ホーム 概要 ヘルプ ログアウト

サーバ: SATOHIROSHI44
インスタンス: CACHE4
ユーザ: UnknownUser

【ホーム】 > 【SQL】 > 【SQLクエリ実行】

データベース

- %SYS
- ARC
- DEMO
- DOCBOOK
- EDU
- HYPER
- ISJSAMPLES
- KB
- PAL
- SAMPLES
- SE
- SQLDEMO
- TEST
- USER**

以下のフォームでデータベース USER で実行するSQLクエリを入力してください:

クエリ実行 | クエリプラン表示 | クエリ履歴 | クエリビルド | 表示: Display Mode | 最大行: 1000

```
select * from SQLUser.KIOM
```

実行されたSQLクエリの結果を以下に表示します: [最終更新: 2008-03-06 14:52:06.495](#)

SQLCODE: 100 列数: 3 パフォーマンス: 0.001 秒 29 グローバル参照

#	ID	昼の気温	地域	日付	夜の気温	夜の気温
1	京都 61061	27	京都	03/06/2008	20	18
2	大塚 61117	25	大塚	05/01/2008	19	21
3	神奈川 61136	24	神奈川	05/20/2008	16	20

完了

ページが表示されました

インデックスをもつ複数の既存グローバルを SQL テーブルにマッピングする方法

以下のようなグローバルをマッピングします。

データ

^PD(患者 ID,"Address")=住所^郵便番号

^PD(患者 ID,"Visits",VisitDate,VisitTime)=症状^費用

インデックス

^PI(名前, 患者 ID)=""

ex)

^PD("012-3434")="山田太郎^39873^03-xxxx-xxxx~090-xxxx-xxxx^InterSystems"

^PD("012-3434","Address")="東京都新宿区^160-xxxx"

^PD("012-3434","Visits",58809,43200)="風邪^2000"

^PD("012-3434","Visits",58909,43300)="骨折^5000"

^PI("山田太郎", "012-3434")=""

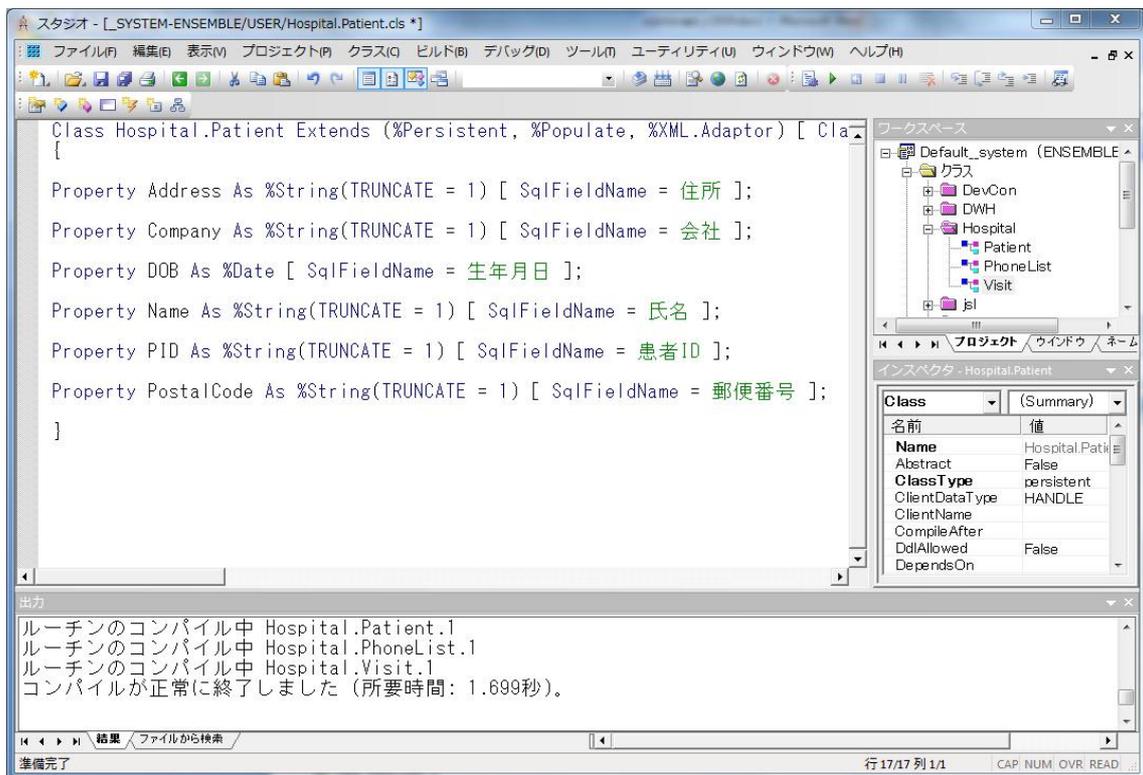
クラス設計

ある患者は複数の電話番号を持ち、複数の病歴を持つ。
患者情報を削除した場合、すべての情報が削除される。

患者クラス

- 患者 ID
- 名前
- 生年月日
- 住所
- 郵便番号
- 会社
- 電話 -(親子関係)- 電話番号クラス
- 病歴 -(親子関係)- 病歴クラス

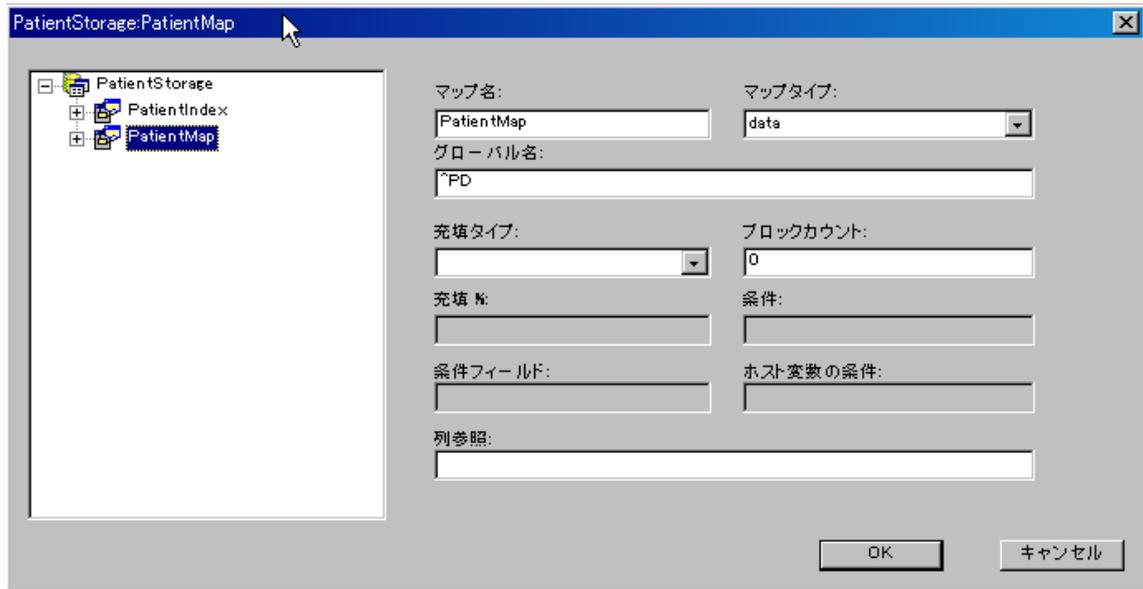
Patient クラスを作成



PID はインデックスとなるので、インデックス定義を追加
Index PIDIndex On PID [IdKey, PrimaryKey, Unique];

新規 Storage を作成する。「CacheSQL ストレージ」を選択。

Inspector から SQL Storage 設定画面を起動する。

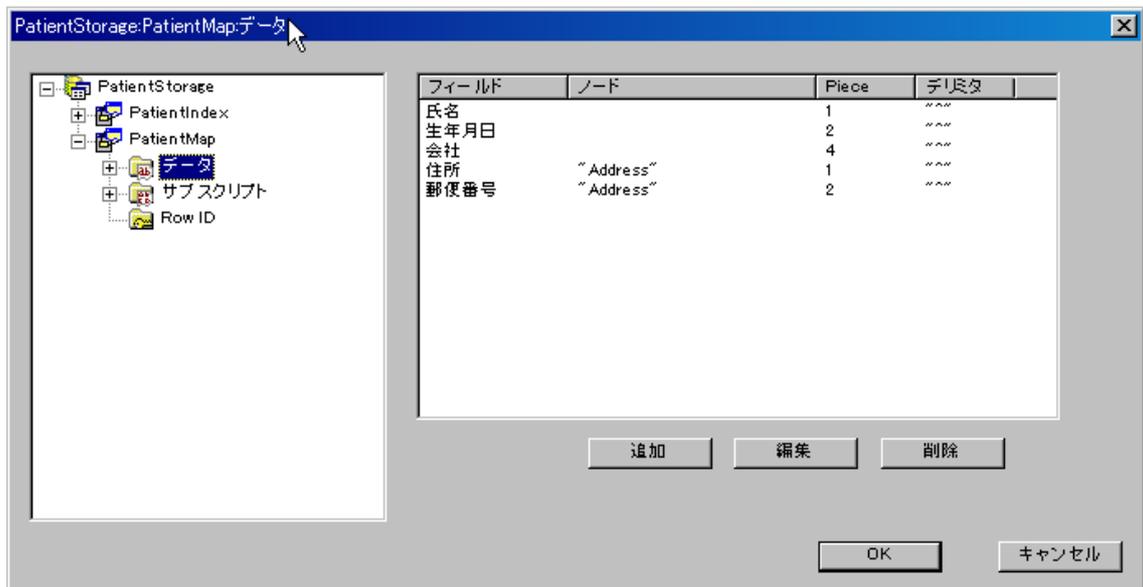


マップ名： PatientMap(任意。スペースは NG)

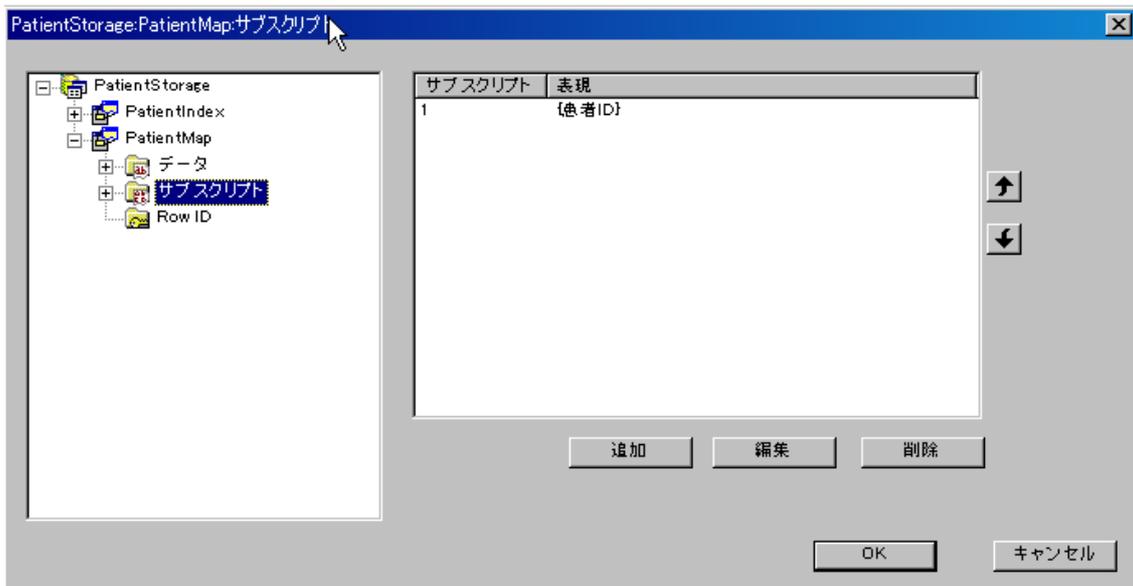
マップタイプ： data

グローバル名： ^PD

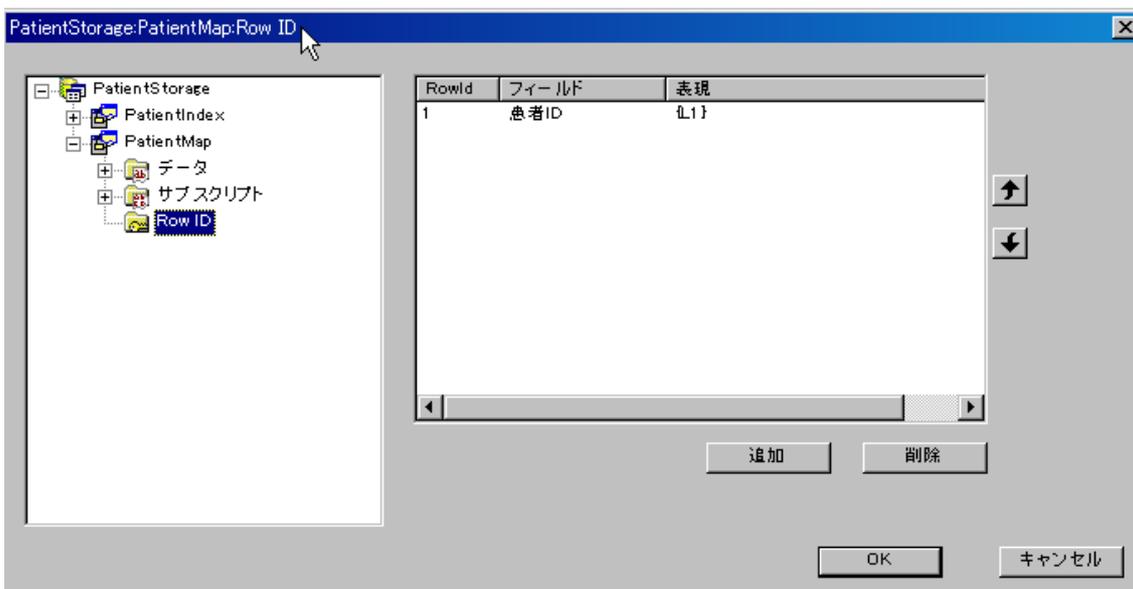
- データ



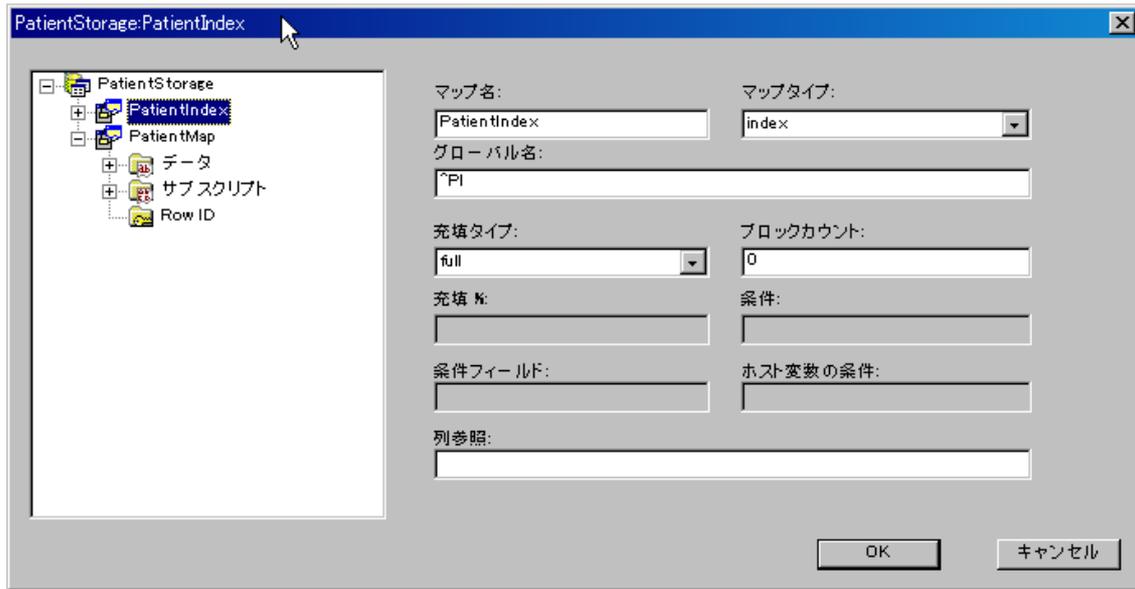
- サブスクリプト



・ RowID

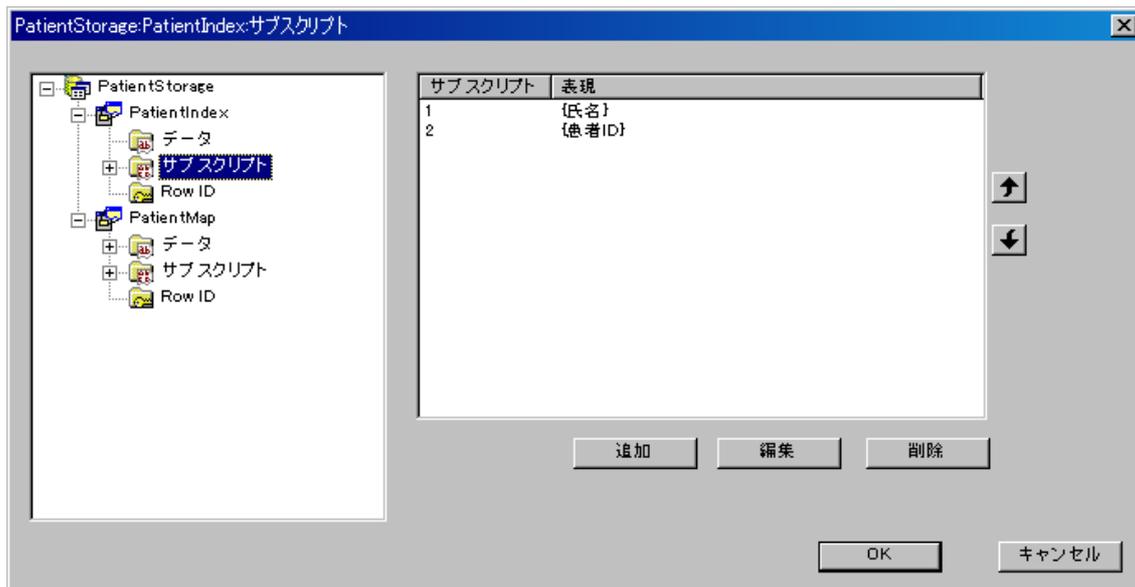


インデックスのマッピングも作成

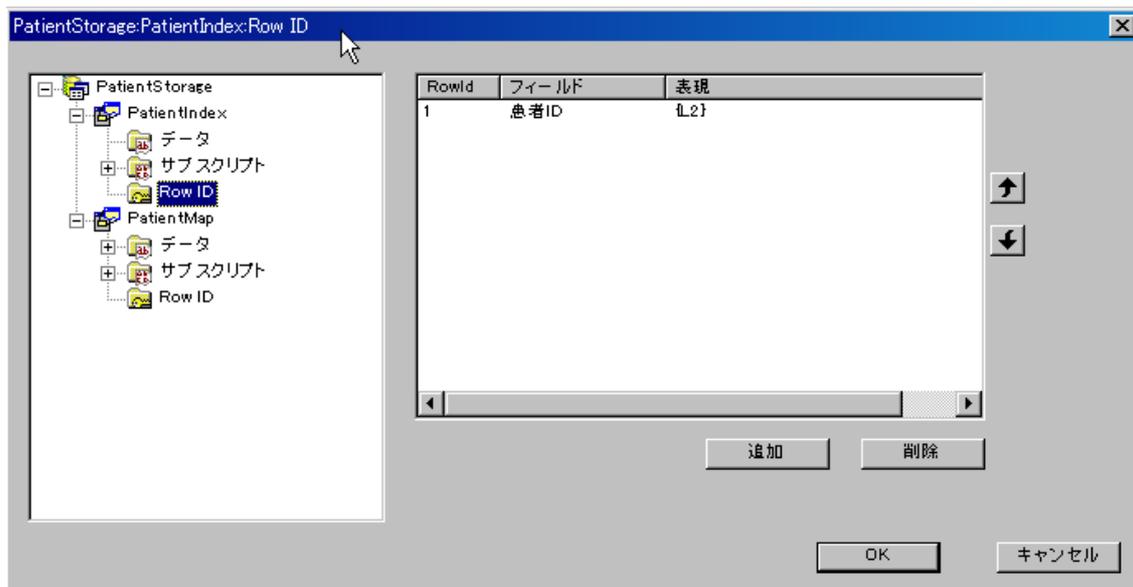


- 充填タイプ：full を選択

- サブスクリプト



- RowID

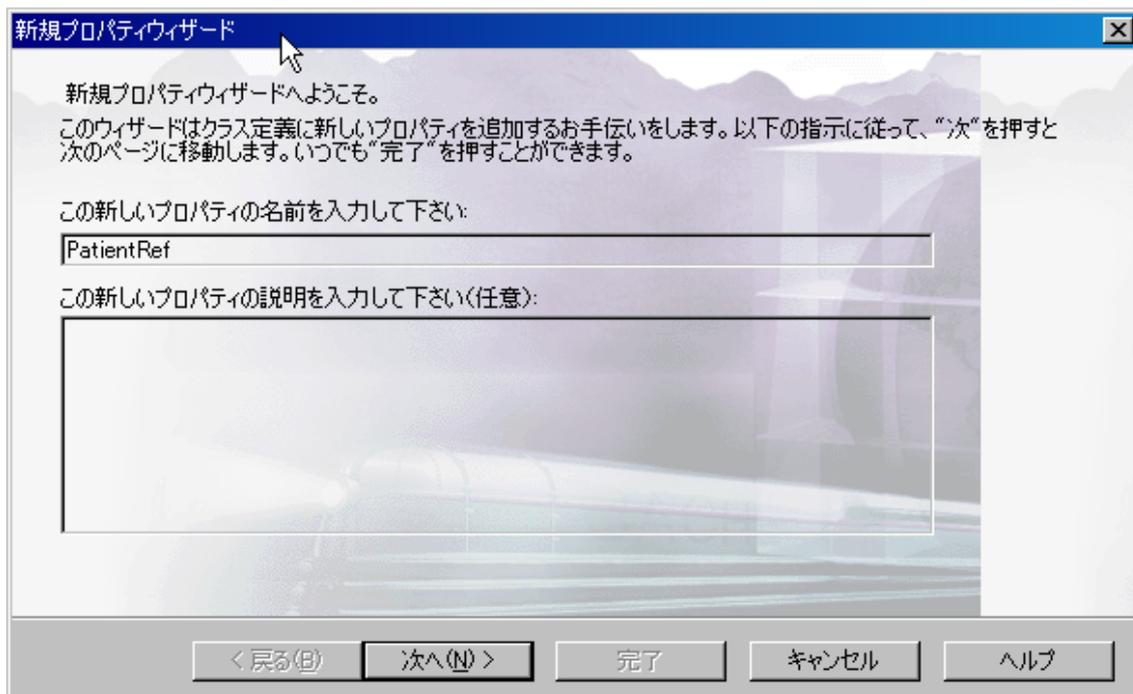


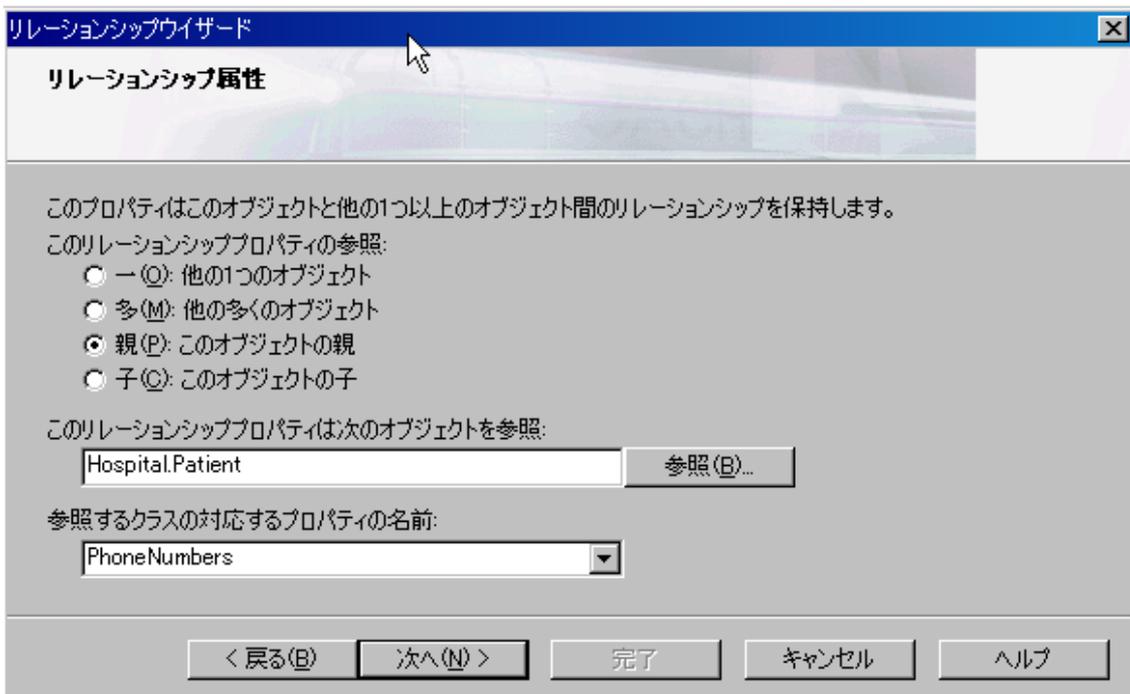
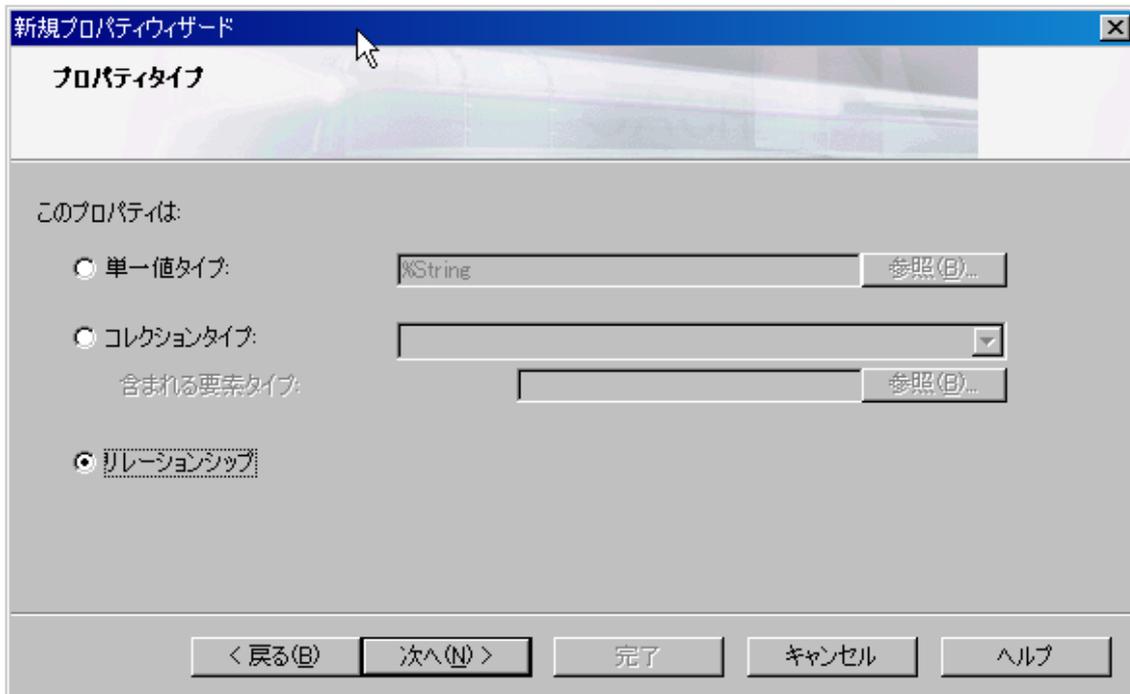
Patient クラスをコンパイルする。

PhoneList クラスを作成

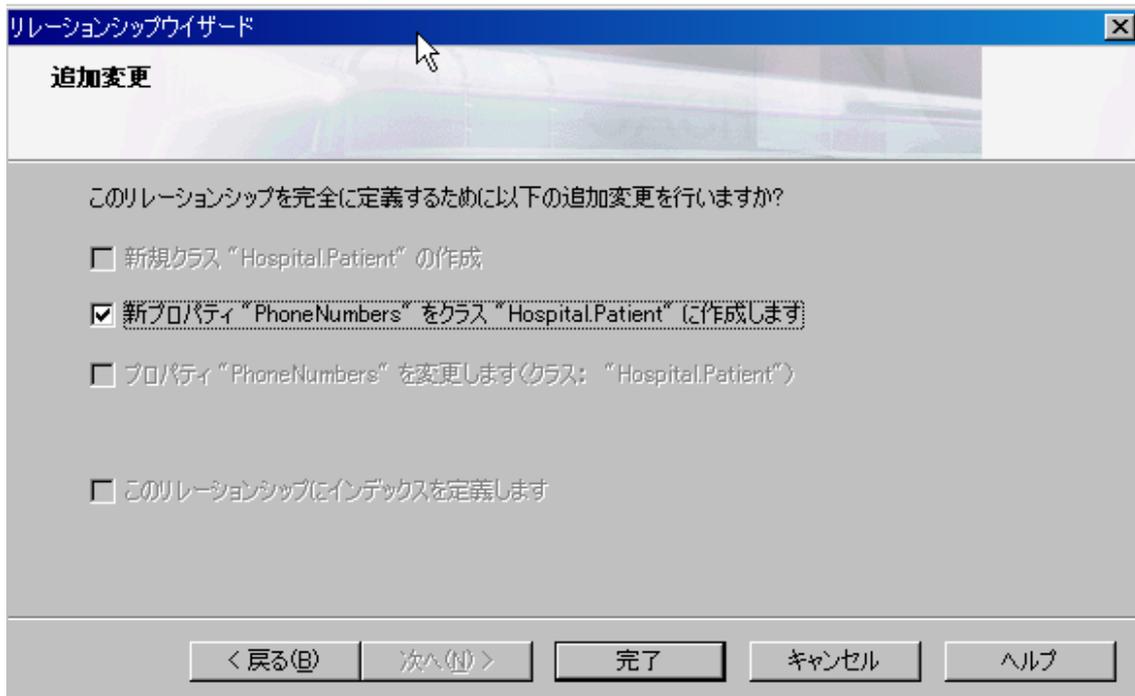
Hostpital.PhoneList クラスを作成。

新規プロパティ PatientRef を追加し、Relationship を定義する





PhoneNumbers はリストに存在しないので、ここで手入力する



- ・ プロパティ Phone, Counter を追加。Counter プロパティは埋め込まれている電話番号の位置を特定するカウンタ。

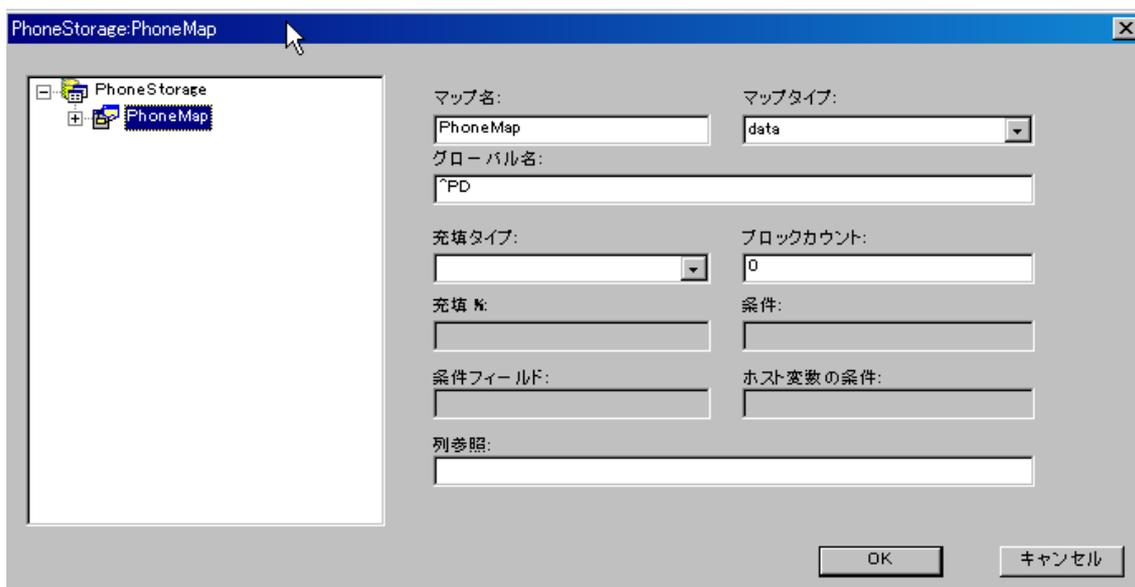
```
Property Counter As %Integer [ SqlFieldName = カウンタ ];
```

```
Property Phone As %String [ SqlFieldName = 電話番号 ];
```

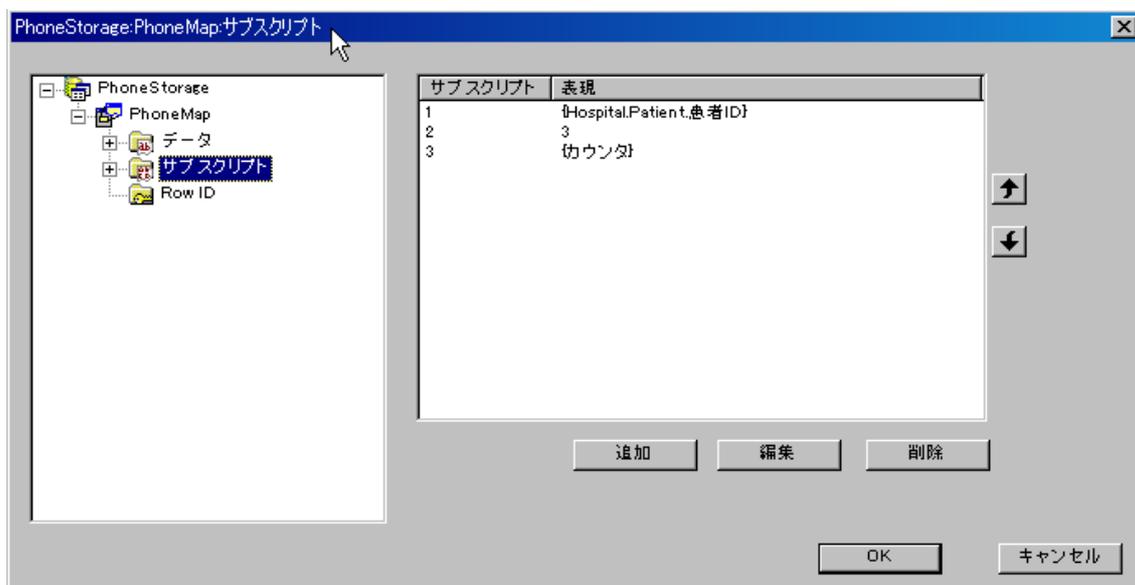
Counter が Index となるので、インデックスの定義も追加する。

```
Index PhoneIndex On Counter [ IdKey, PrimaryKey, Unique ];
```

- ・ 新規ストレージ PhoneStorage を追加し、変更する



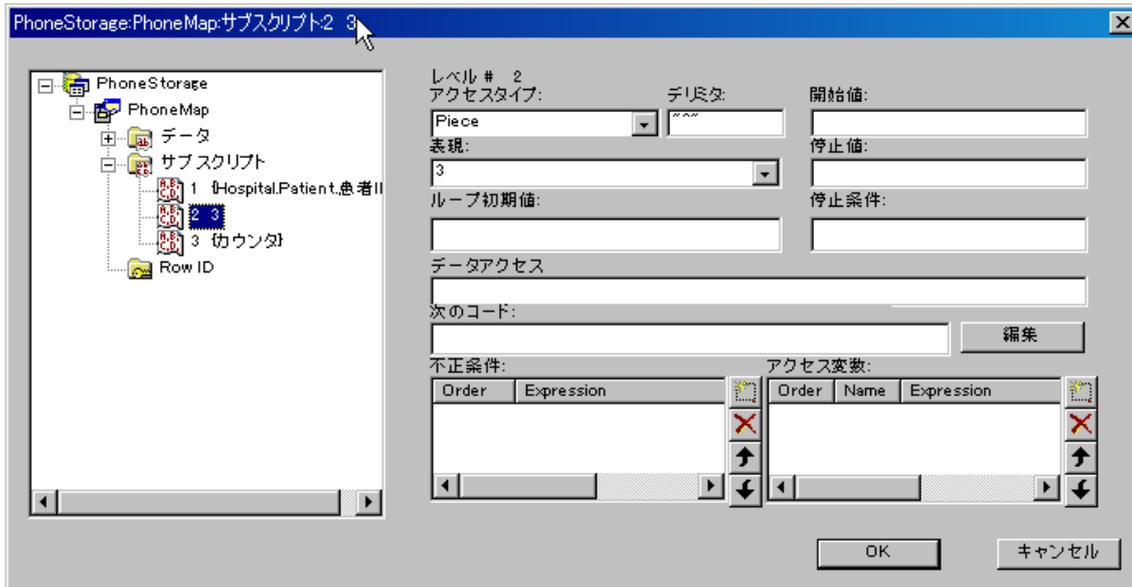
- ・ サブスクリプト



{Hospital.Patient.患者 ID}はリストには存在しないので、直接入力

- ・ サブスクリプト Level2,3 ではアクセスタイプ、デリミタをそれぞれ設定する

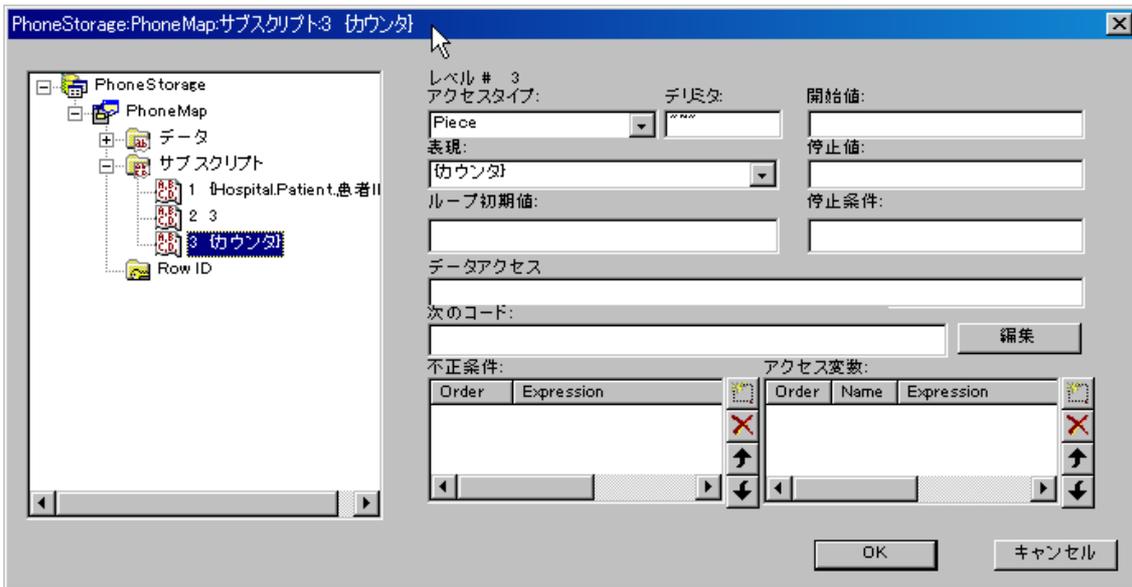
L2



アクセスタイプ : Piece

デリミタ : “^”

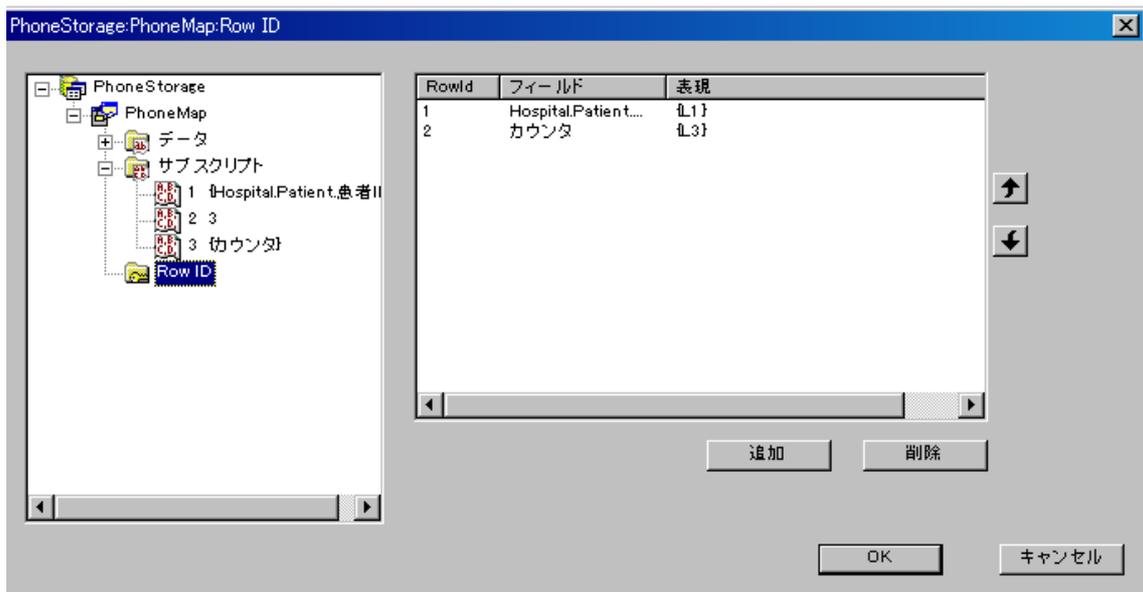
L3



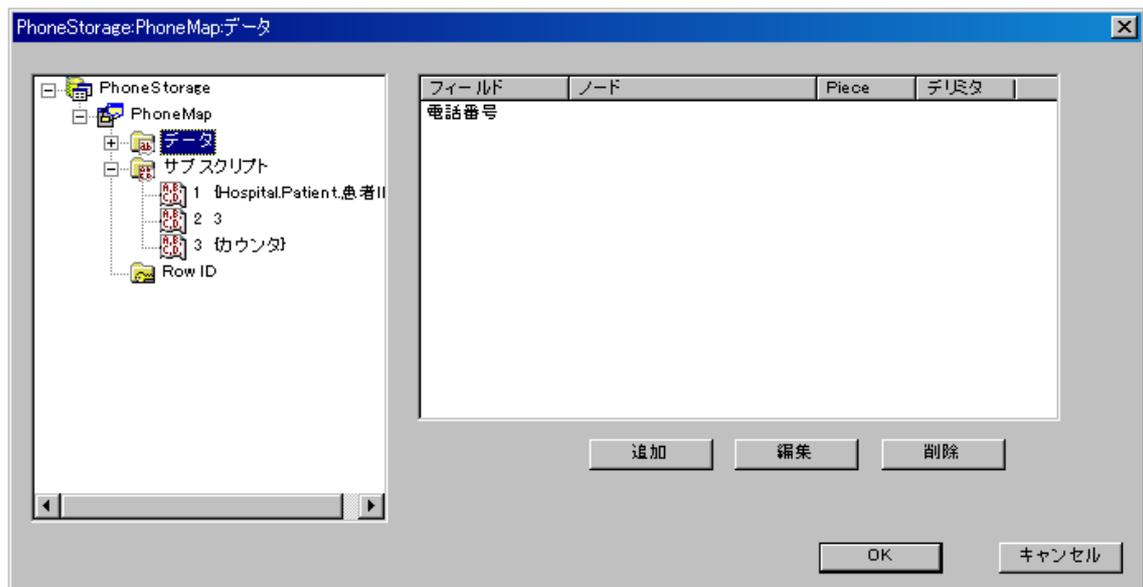
アクセスタイプ : Piece

デリミタ : “~”

- RowID

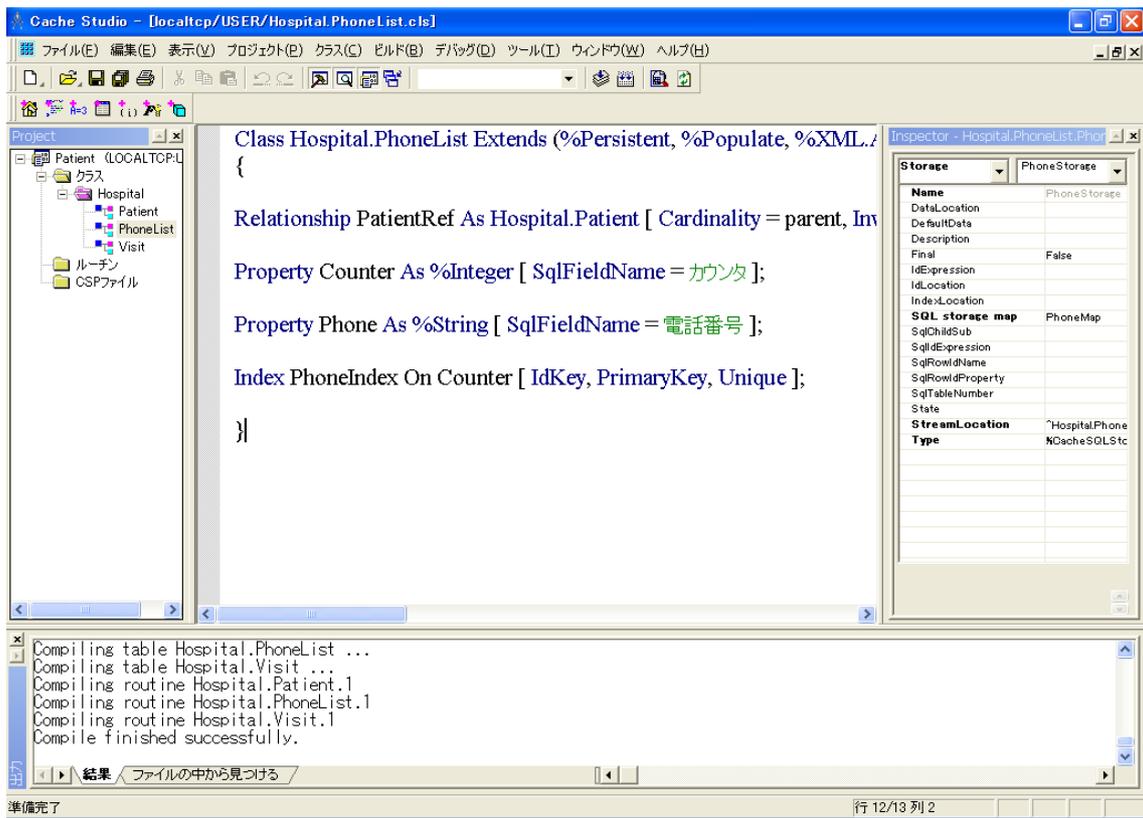


- Data



「電話番号」とだけ追加。他の設定は必要ありません。

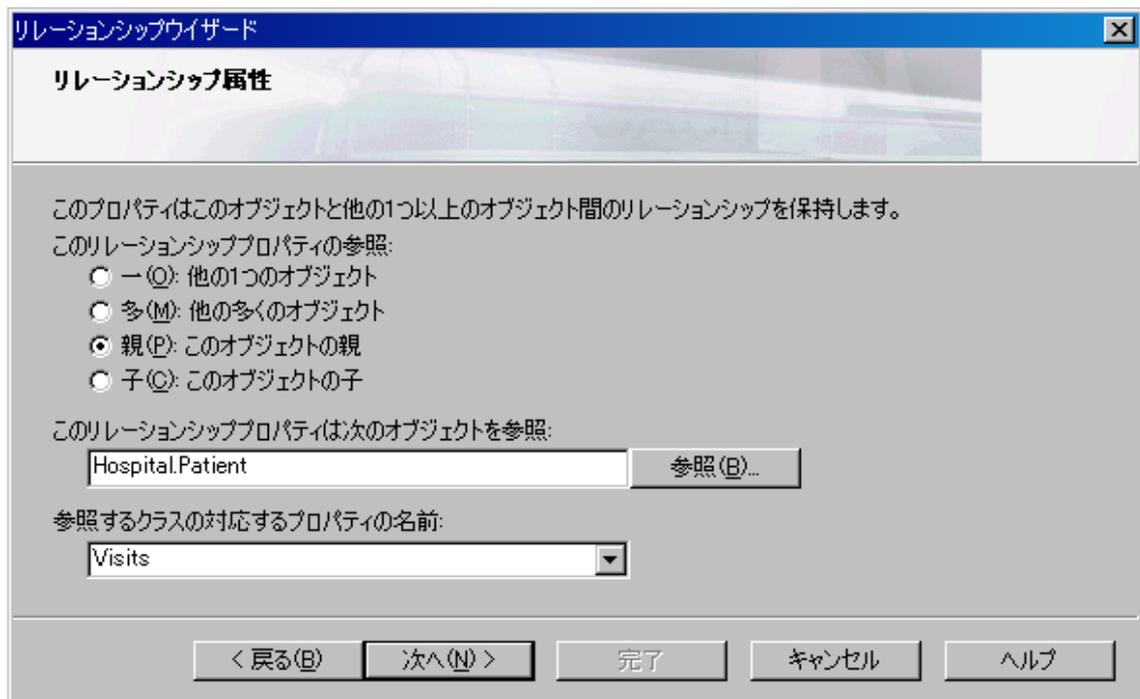
クラス PhoneList をコンパイル



Visit クラスを作成

Hostpital. Visit クラスを作成。

先ほどと同様に新規プロパティ PatientRef を追加し、Relationship を定義する



Visits はリストに存在しないので、ここで手入力する

それ以外のプロパティは

```
Property Payment As %Currency [ SqlFieldName = 費用 ];
```

```
Property Symptom As %String [ SqlFieldName = 症状 ];
```

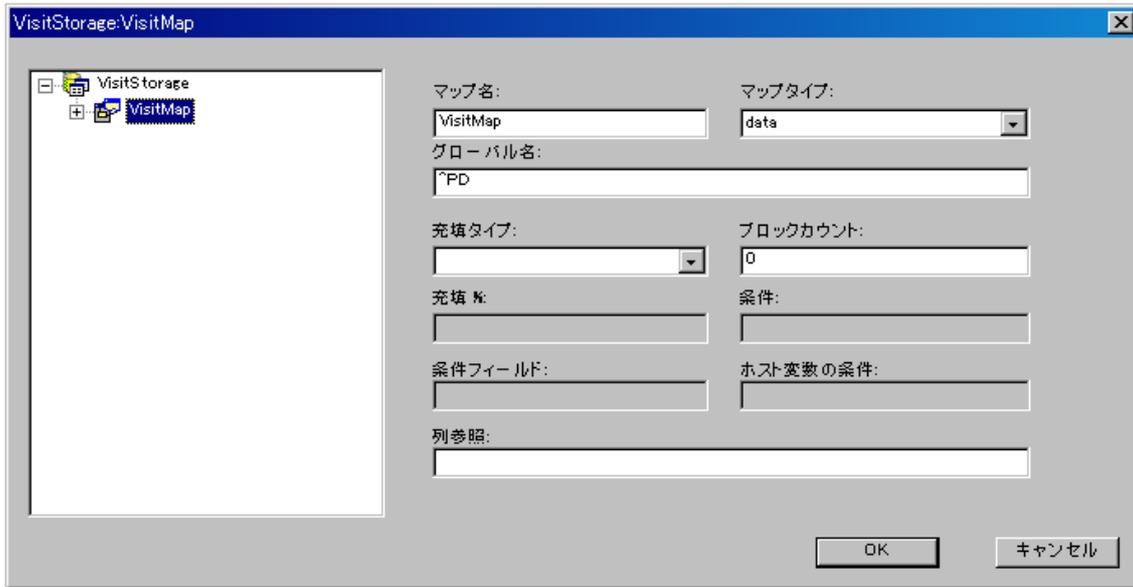
```
Property VisitDate As %Date [ SqlFieldName = 受診日 ];
```

```
Property VisitTime As %Time [ SqlFieldName = 受診時間 ];
```

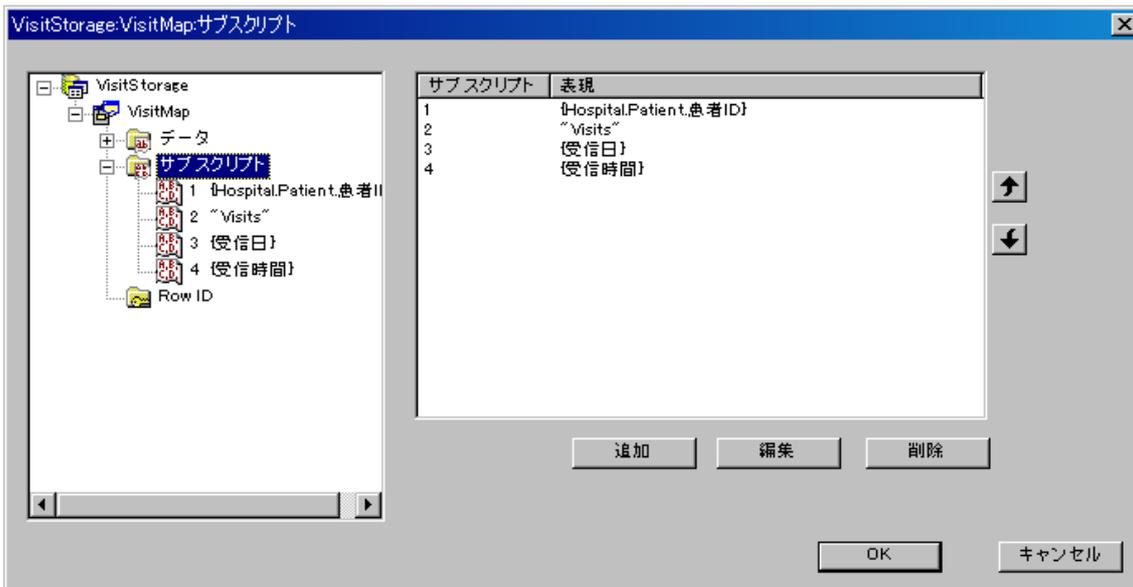
VisitDate, VisitTime に対するインデックスの定義を追加する。

```
Index VisitIndex On (VisitDate, VisitTime) [ IdKey, PrimaryKey, Unique ];
```

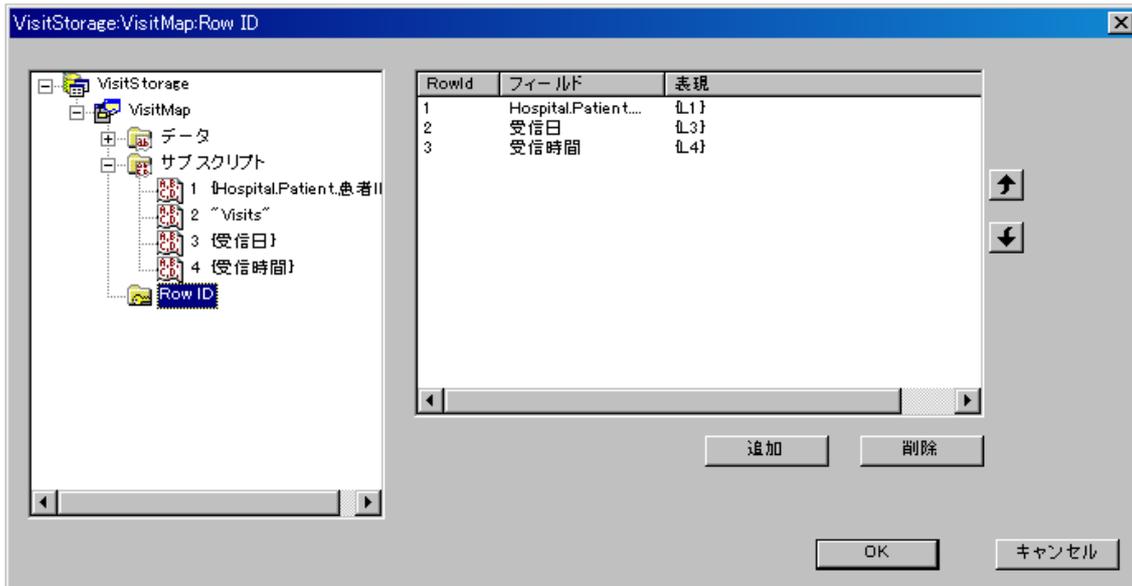
- 新規ストレージ、VisitStorage を作成し、変更する



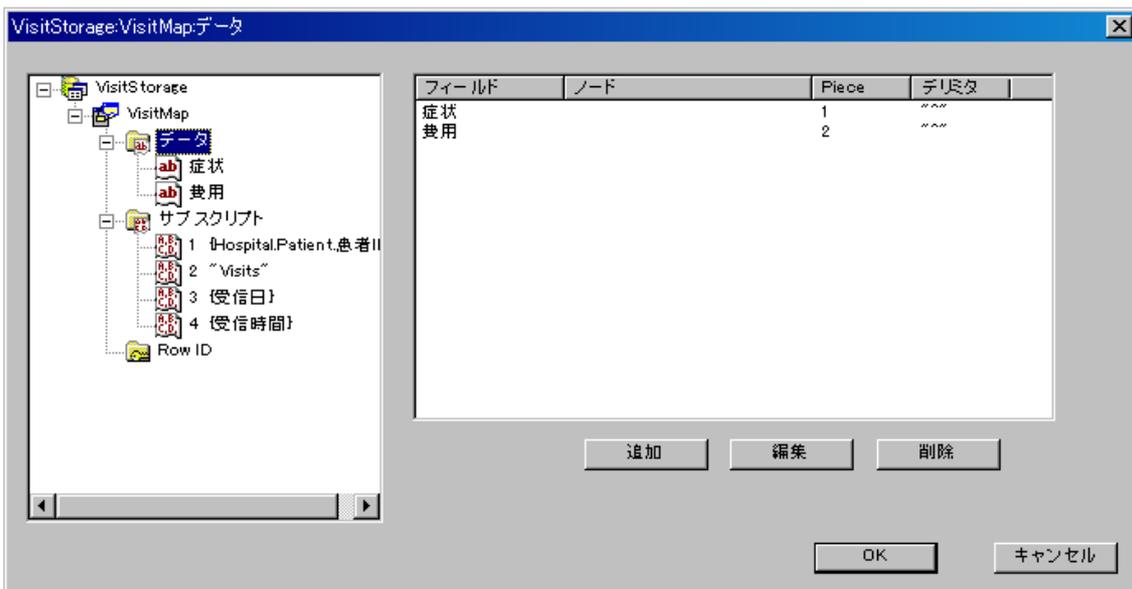
- サブスクリプト



- RowID



・ データ



クラス Visit をコンパイル

Cache Studio - [localtcp/USER/Hospital.Visit.cls]

ファイル(E) 編集(E) 表示(V) プロジェクト(P) クラス(C) ビルド(B) デバッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)

Project: Patient (LOCALTCP)

- クラス
 - Hospital
 - Patient
 - PhoneList
 - Visit
 - ルーチン
 - CSPファイル

```

Class Hospital.Visit Extends (%Persistent, %Populate, %XML.Ada
{
  Index VisitIndex On (VisitDate, VisitTime) [ IdKey, PrimaryKey, U
  Property Payment As %Currency [ SqlFieldName = 費用 ];
  Property Symptom As %String [ SqlFieldName = 症状 ];
  Relationship PatientRef As Hospital.Patient [ Cardinality = parent, ]
  Property VisitDate As %Date [ SqlFieldName = 受信日 ];
  Property VisitTime As %Time [ SqlFieldName = 受信時間 ];
}
  
```

Inspector - Hospital.Visit.VisitStorage

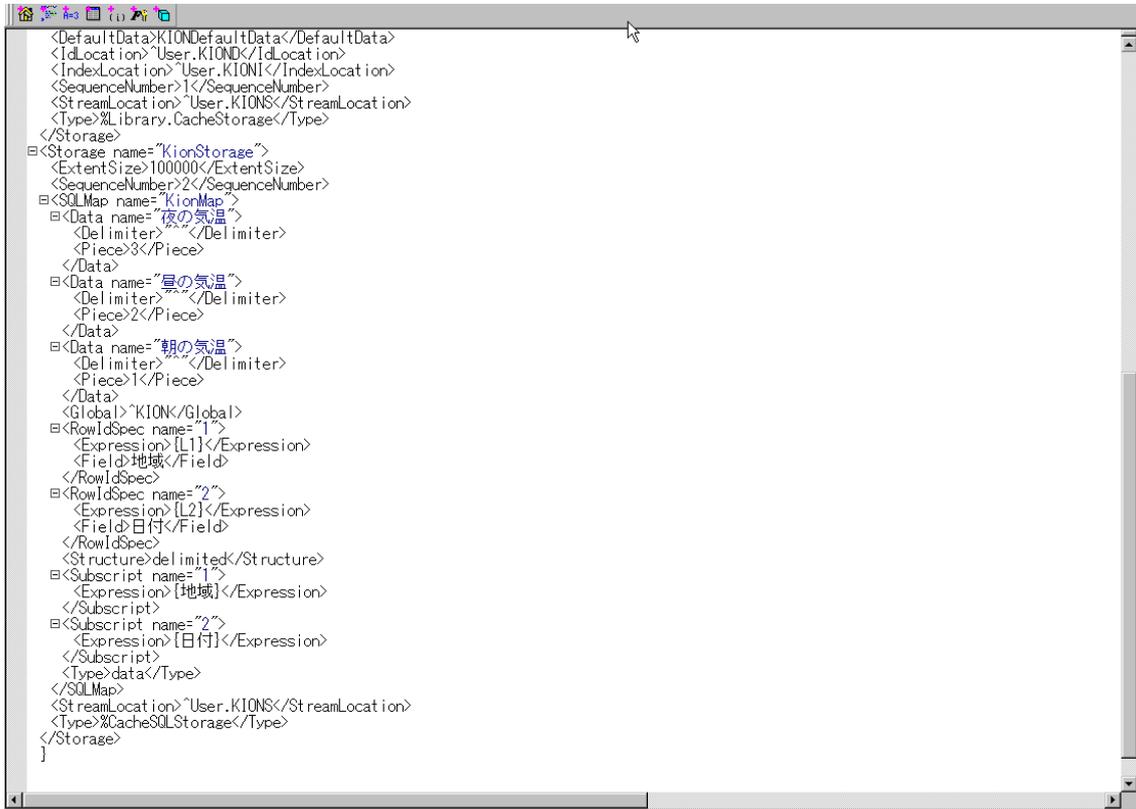
Storage	VisitStorage
Name	VisitStorage
DataLocation	
DefaultData	
Description	
Final	False
IdExpression	
IdLocation	
IndexLocation	
SQL storage map	VisitMap
SqlChildSub	
SqlIdExpression	
SqlRowIdName	
SqlRowIdProperty	
SqlTableName	
State	
StreamLocation	^Hospital.VisitS
Type	NCacheSQLStc

Compiling table Hospital.PhoneList ...
 Compiling table Hospital.Visit ...
 Compiling routine Hospital.Patient.1
 Compiling routine Hospital.PhoneList.1
 Compiling routine Hospital.Visit.1
 Compile finished successfully.

準備完了 行 5/17 列 1

ストレージ XML 定義を直接編集する方法

編集ペインに以下の様なストレージ定義が XML 形式で表示されます。



```
<DefaultData>KIONDefaultData</DefaultData>
<IdLocation>"User.KION</IdLocation>
<IndexLocation>"User.KIONI</IndexLocation>
<SequenceNumber>1</SequenceNumber>
<StreamLocation>"User.KIONS</StreamLocation>
<Type>%Library.CacheStorage</Type>
</Storage>
<Storage name="KionStorage">
  <ExtentSize>100000</ExtentSize>
  <SequenceNumber>2</SequenceNumber>
  <SQLMap name="KionMap">
    <Data name="夜の気温">
      <Delimiter>"</Delimiter>
      <Piece>3</Piece>
    </Data>
    <Data name="昼の気温">
      <Delimiter>"</Delimiter>
      <Piece>2</Piece>
    </Data>
    <Data name="朝の気温">
      <Delimiter>"</Delimiter>
      <Piece>1</Piece>
    </Data>
    <Global>"KION</Global>
    <RowIdSpec name="1">
      <Expression>[L1]</Expression>
      <Field>地域</Field>
    </RowIdSpec>
    <RowIdSpec name="2">
      <Expression>[L2]</Expression>
      <Field>日付</Field>
    </RowIdSpec>
    <Structure>delimited</Structure>
    <Subscript name="1">
      <Expression>[地域]</Expression>
    </Subscript>
    <Subscript name="2">
      <Expression>[日付]</Expression>
    </Subscript>
    <Type>data</Type>
  </SQLMap>
  <StreamLocation>"User.KIONS</StreamLocation>
  <Type>%CacheSQLStorage</Type>
</Storage>
}
```

この XML 定義をスタジオのペイン上で編集することも可能ですし、クラス定義を XML ファイルとしてエクスポートし、他のエディタで編集することも可能です。