



Работа с существующими глобалами через объекты и SQL

Вадим Федоров

InterSystems Corporation

Содержание



Обзор стратегий хранения

CacheStorage

CacheSQLStorage

CustomStorage

Пример CacheSQLStorage

Сравнение стратегий хранения

	CacheStorage	CacheSQLStorage	CustomStorage
SQL	✓	✓	□
Objects	✓	✓	□

✓ Обеспечивается Caché

□ Реализуется разработчиком

Выбираем стратегию хранения

- CacheStorage идеально подходит для новых приложений
- CacheSQLStorage применяется, когда с существующими глобалами можно и нужно работать с помощью SQL
- CustomStorage используется тогда, когда нельзя работать с глобалами через SQL и нужно реализовывать собственную сложную логику для обеспечения объектного доступа

Содержание



Обзор стратегий хранения



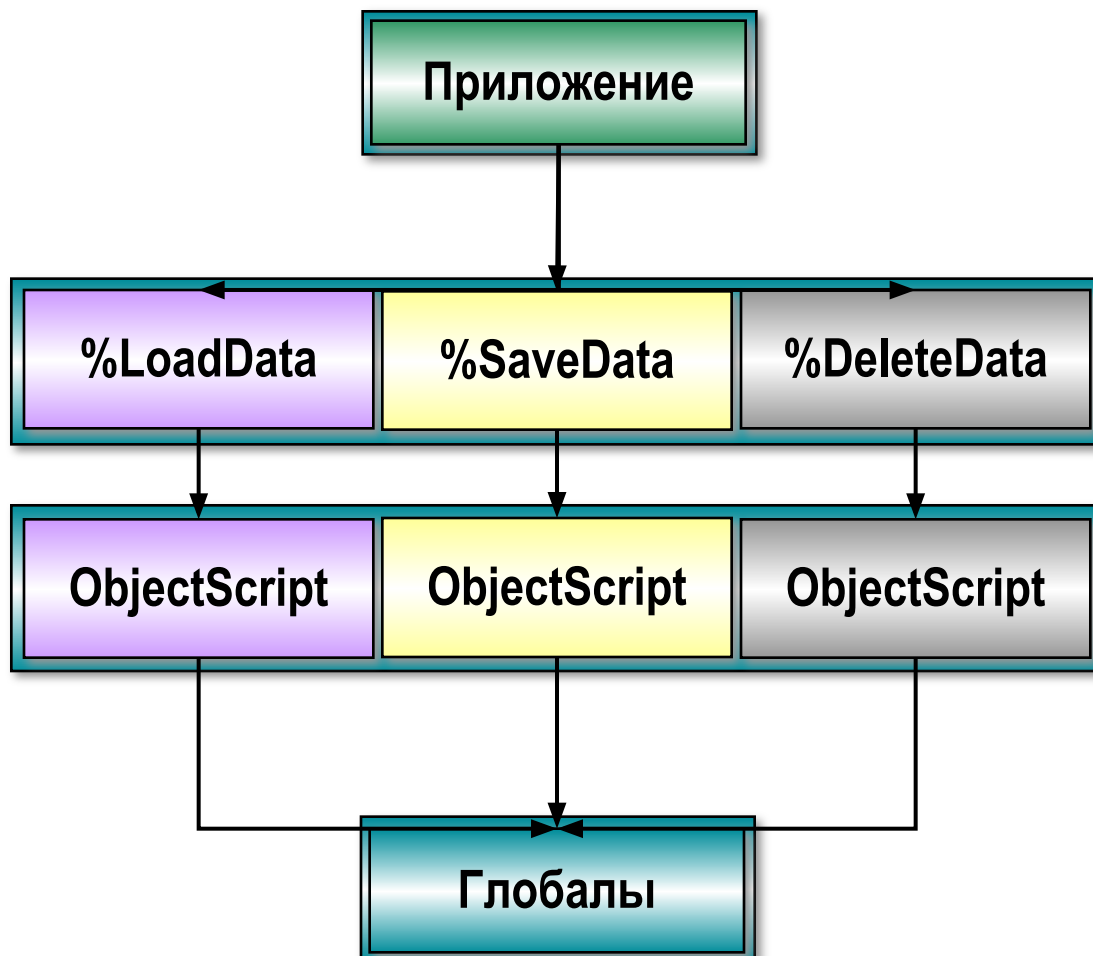
CacheStorage

CacheSQLStorage

CustomStorage

Пример CacheSQLStorage

Обзор CacheStorage



Объектное API

Объектная
реализация

Информация о CacheStorage

- CacheStorage генерирует глобалы, в которых используется \$ListBuild
- Уникальный идентификатор (IDKey / PrimaryKey) может автоматически сгенерирован Cache или задан разработчиком вручную
 - Это влияет на структуру глобалов
- Можно управлять хранением свойств классов в глобалах
 - Это влияет на структуру значений глобалов

Содержание



Обзор стратегий хранения



CacheStorage

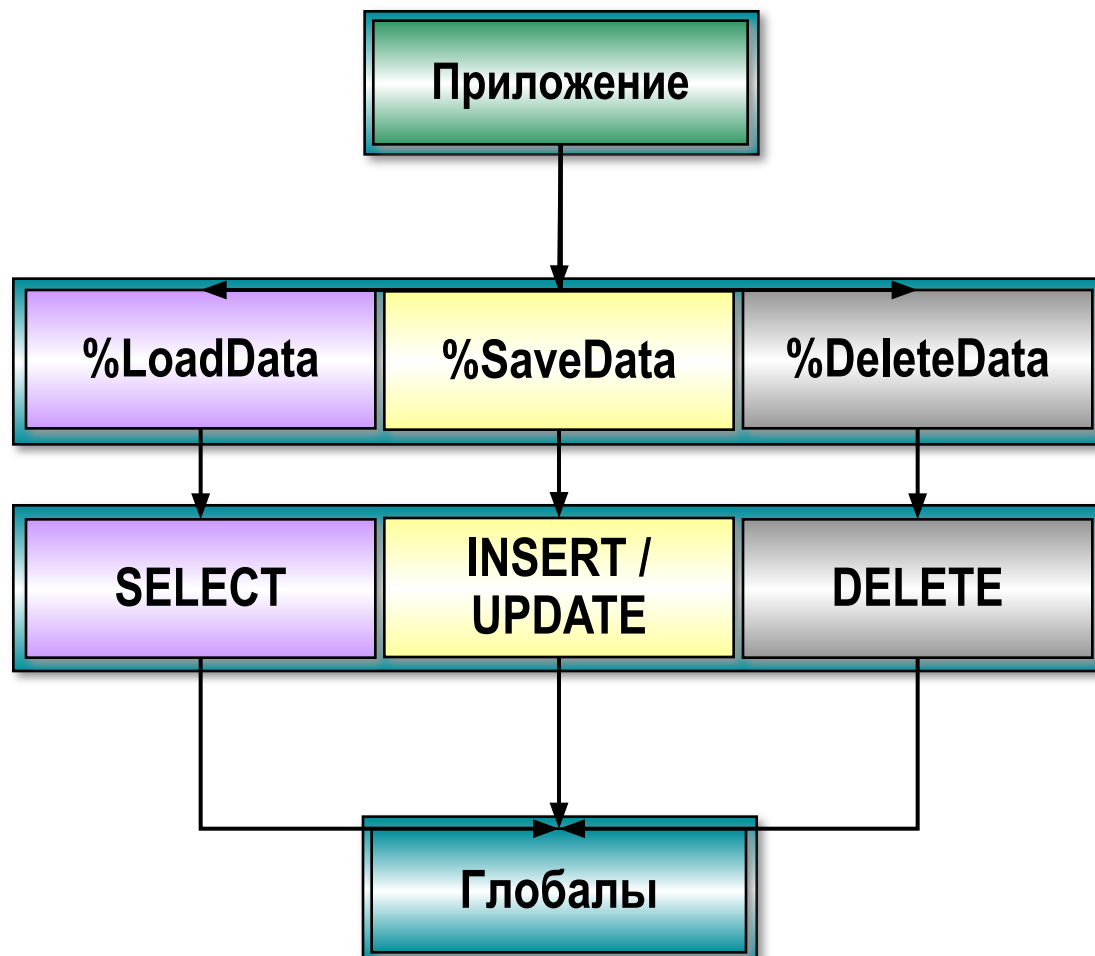


CacheSQLStorage

CustomStorage

Пример CacheSQLStorage

Обзор CacheSQLStorage



Объектное API

SQL
Реализация

Обзор CacheSQLStorage

- Создайте Persistent-класс
- Добавьте свойства в класс
- Определите свойство (свойства), которое будет идентификатором класса (IDKey / Primary Key)
- Создайте стратегию хранения выставив соответствие между свойствами класса и данными глобала

Особенности CacheSQLStorage

- При работе через объекты будут вызываться триггеры (при использовании CacheStorage они не вызываются)!
- Поддерживается ссылка и Parent-Child отношение для связи классов

Создание CacheSQLStorage

- CacheSQLStorage обычно создается:
 - Программистом
 - Программой конвертации из F-DBMS
 - Программой конвертации из KB-SQL

Обеспечение SQL-доступа

- Не ко всем структурам глобалов можно настроить CacheSQLStorage так чтобы обеспечить полный SQL-доступ (Read/Write/Delete)
- Некоторые структуры подходят только для чтения через SQL (SELECT)
- Чтобы обеспечить обновление иногда необходима дополнительная работа, кроме установки Mapping

Виды карт

- Различают следующие виды CacheSQLStorage карт:
 - Данные (MasterMap): Должны быть определены все поля
 - Индексы: Должны быть определены некоторые поля
 - Full (по умолчанию): Все данные попадают в индексы
 - Conditional: Данные попадают в индексы, если выполняется
 - Nonnull: Нулевые значения (Null values) не попадают в индексы

ID и Primary Key

- IDKey индексы определяют уникальные идентификаторы для объектов
- Primary Key индексы определяют уникальные идентификаторы для SQL
- IDKey и Primary Key индексы обычно строятся по одним и тем же полям

Индексы глобалов (Subscripts)

- Индексы (subscripts) карт обычно эквивалентны индексам глобалов
- Индексы карт используется для формирования кода, перебирающего записи таблицы.
- Поддерживаются следующие типы индексов (subscripts) :
 - Sub: Основанный на «стандартном» индексе глобалов
 - Piece: Основанный на определенной позиции (используется разделитель)
 - Global: Основанный на данных, хранящихся в нескольких глобалах
 - Other: Основанный на пользовательском коде

RowID

- Существует тесная связь между IDKey и RowID для карт данных, но не для карт индексов
- А RowID используется для уникальной идентификации данных в глобале, на основе индекса глобала, определенного в карте
- Например, для глобала:
 $\wedge \text{Person}(\text{PersonID}, \text{"Cars"}, \text{CarID}) = \text{"Make}^{\wedge} \text{Model}^{\wedge} \text{Year}"$
2 поля будут нашими RowID:
PersonID: хранится на первом уровне {L1}
CarID: хранится на третьем уровне {L3}

Карты данных

- Когда определены индексы глобала, нужно определить хранение свойств класса в глобале
- Можно определить дополнительные узлы индекса глобала (только литералы)
- Можно использовать \$Piece или \$ListBuild

Подробности редактирования карт

- Map Name: Имя карты.
- Map Type: Данные или Индексы (Data or Index)
- Global Name: Имя глобала (^...) или локального массива
- Node Structure: Структура узла: \$Piece или \$List
- Population Type: Тип заполнения
- Population %: Оценка предполагаемого количества рядов в индексе
- Condition: Выражение определяющее условие. Например, {Name} '=""
- Conditional Fields: Поля, по которым будет проверяться условие
- Conditional with hostvars: Булева значение, которое влияет на использование индекса кэшированными запросами
- Row Reference: Позволяет программистам переопределить сгенерированный RowID

Подробности редактирования индексов глобалов

- Access Type: Тип доступа. Sub, Piece, Global или Other
- Delimiter: Разделитель. Используется, если тип доступа Piece
- Expression: Выражение. Обычно {поле}, "string" или число, или определенная позиция
- Loop Init Value: Не включаемое значение, используемое для генерации кода обхода
- Start Value: Включаемое значение, используемое в сгенерированном коде обхода
- Stop Value: Значение, при котором обход останавливается
- Stop Expression: Выражение, при котором обход останавливается, например, {L1}>200
- Data Access: Доступ к данным. Переопределяет контекст текущего выражения для вычисления значения текущего уровня доступа (Override the context of the current access-level's value expression)
- Next Code: Используется программистом для переопределения генерируемого кода обхода
- Invalid Conditions: Выражения, используемые для исключения рядов из карты. Например, {L1}<1
- Access Variables: Переменные, используемые программистом, для обеспечения уникальности имен

Подробности редактирования RowID

- RowID: Позиция поля в спецификации RowID
- Field: Имя поля, составляющего часть RowID
- Expression: Уровень внутри определения индексов глобала (subscript definition). Например, {L2} или {L6}

Подробности редактирования данных

- Field: Имя поля
- Node: Дополнительный индекс глобала (только литерал), где находится поле
- Piece: Позиция в строке \$Piece
- Delimiter: Разделитель. Например, “^” или \$c(1)

Содержание



Обзор стратегий хранения



CacheStorage



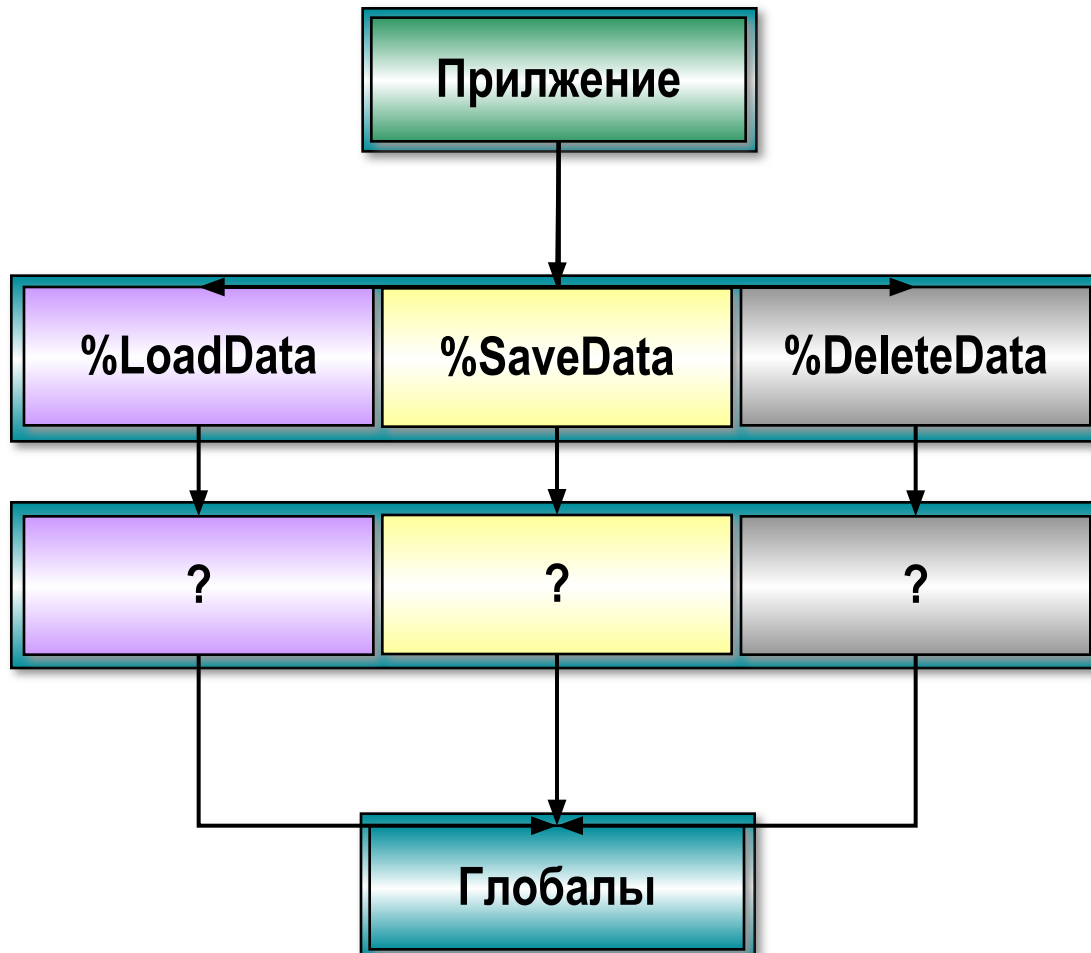
CacheSQLStorage



CustomStorage

Пример CacheSQLStorage

Обзор CustomStorage



Объектные API

Собственная
реализация

Обзор CustomStorage

- Создайте Persistent-класс
- Добавьте свойства в класс
- Определите свойство (свойства), которое будет идентификатором класса (IDKey / Primary Key)
- Создайте стратегию хранения выставив соответствие между свойствами класса и данными глобала
- Реализуйте код доступа к объектам: %LoadData, %SaveData, %DeleteData

CustomStorage и SQL

- Для того чтобы использовать SQL с CustomStorage, **необходимо** определить специальный параметр класса:
Parameter SQLENABLED = 1;
- Mapping the SQL portion with CustomStorage is identical to the methods used for CacheSQLStorage

CustomStorage и объекты

- Для того чтобы использовать объекты с CustomStorage, **необходимо** выполнить следующее:
 - Реализовать %LoadData, %SaveData, %DeleteData
 - В Вашем коде Вы должны управлять :
 - ID объектов на диске и в памяти (с помощью метода %IdSet())
 - Переменными экземпляров свойств (имена свойств имеют первые символы “i%”)
 - Concurrency
 - Уникальностью данных
 - Ограничениями, накладываемыми внешних ключей (Foreign key constraints)

%LoadData

- Код, реализованный в %LoadData(), будет выполняться каждый раз, когда загружается объект, обычно после вызова %Open() и %OpenId()
- Пример %LoadData:

```
Method %LoadData(id As %Library.String) As %Library.Status
{
    Set i%SSN = id
    Set i%Name = $Piece(^P(id),"^",1)
    Set i%DOB = $Piece(^P(id),"^",2)

    Quit $$$OK
}
```

%SaveData

- Код, реализованный в %SaveData(), будет выполняться каждый раз, когда сохраняется объект, в результате вызова метода %Save()
- Пример %SaveData:

```
Method %SaveData(id As %Library.String) As %Library.Status
{
    Lock ^P(id):5 If '$Test Quit $$$ERROR($$$LockFailedToAcquireExclusive)

    Set id = i%SSN
    Do ..%IdSet(id)

    Set $Piece(^P(id),"^",1) = i%Name
    Set $Piece(^P(id),"^",2) = i%DOB

    Quit $$$OK
}
```

%DeleteData

- Код, реализованный в %DeleteData, будет выполняться каждый раз, когда объект будет удаляться, в результате вызова %Delete() или %DeleteId()
- Пример %DeleteData:

```
Method %DeleteData(id As %String, concurrency as %Integer) As %Status
{
    Lock ^P(id):5 If '$Test Quit $$$ERROR($$$LockFailedToAcquireExclusive)

    Kill ^P(id)

    Quit $$$OK
}
```

Содержание



Обзор стратегий хранения



CacheStorage



CacheSQLStorage

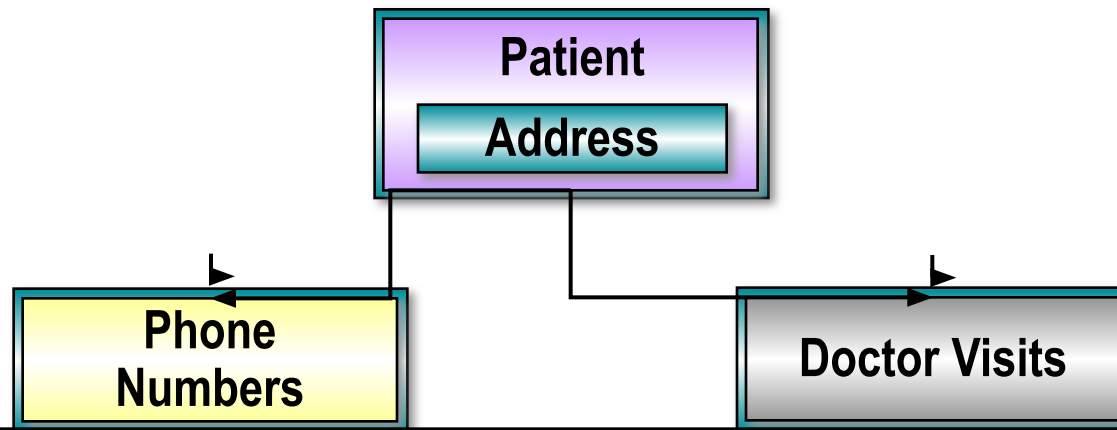


CustomStorage



Пример CacheSQLStorage

Пример модели данных



- Есть два отношения Parent-Children:
 - Пациент может иметь несколько номеров телефона
 - Пациент может посещать доктора несколько раз
- Удаление пациента удаляет его номера телефонов и визиты к врачу

Пример структуры данных глобала

$\text{^P(SSN)} = \text{"Name^DOB^Phone1~Phone2~...~PhoneN^Company"}$

$\text{^P(SSN, "Address")} = \text{"City^PostalCode^Country"}$

$\text{^P(SSN, "Visits", VisitDate, VisitTime)} = \text{"Symptom^Payment"}$

$\text{^P("211-22-1222")} = \text{"Smith,John^39873^718-317-3312~917-225-2213^AT\&T"}$

$\text{^P("211-22-1222", "Address")} = \text{"New York^10312^USA"}$

$\text{^P("211-22-1222", "Visits", 58809, 43200)} = \text{"Cough^15.00"}$

$\text{^P("211-22-1222", "Visits", 58820, 57900)} = \text{"Sore Throat^50.00"}$

Пример структуры индексов глобала

$\text{^PI}(\text{Name,SSN}) = \text{“”}$

$\text{^PI}(\text{“Smith,John”,“211-22-1222”}) = \text{“”}$

Создаем Persistent-класс



New Class Wizard

Welcome to the New Class Wizard.
This wizard will guide you through creating a new Cache class definition.
Please follow the instructions below, pressing "Next" to move on to the next page.
You may press "Finish" at any time.

Enter a package name:

Training

Enter a class name:

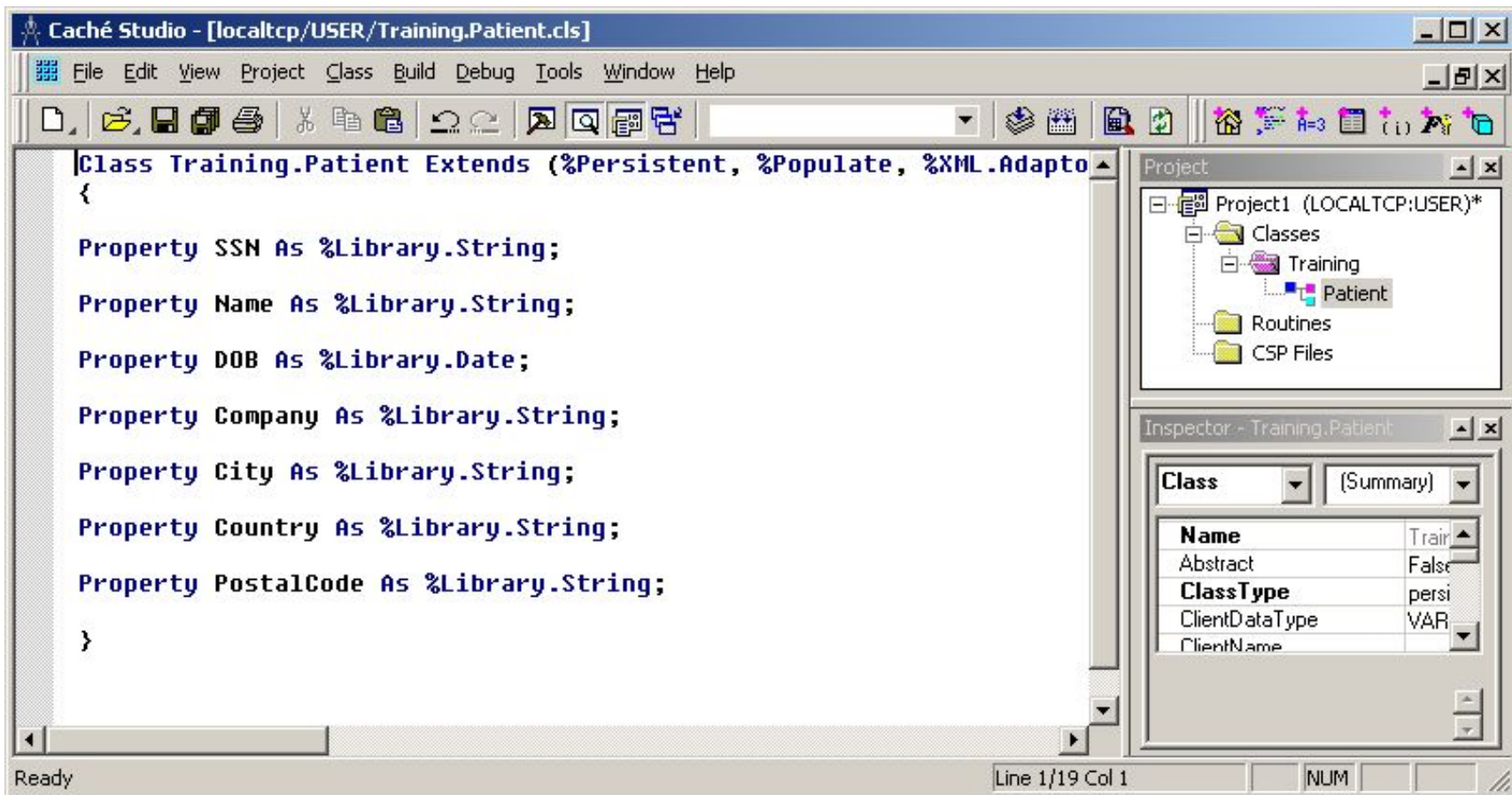
Patient

Enter a description of this new class (optional):

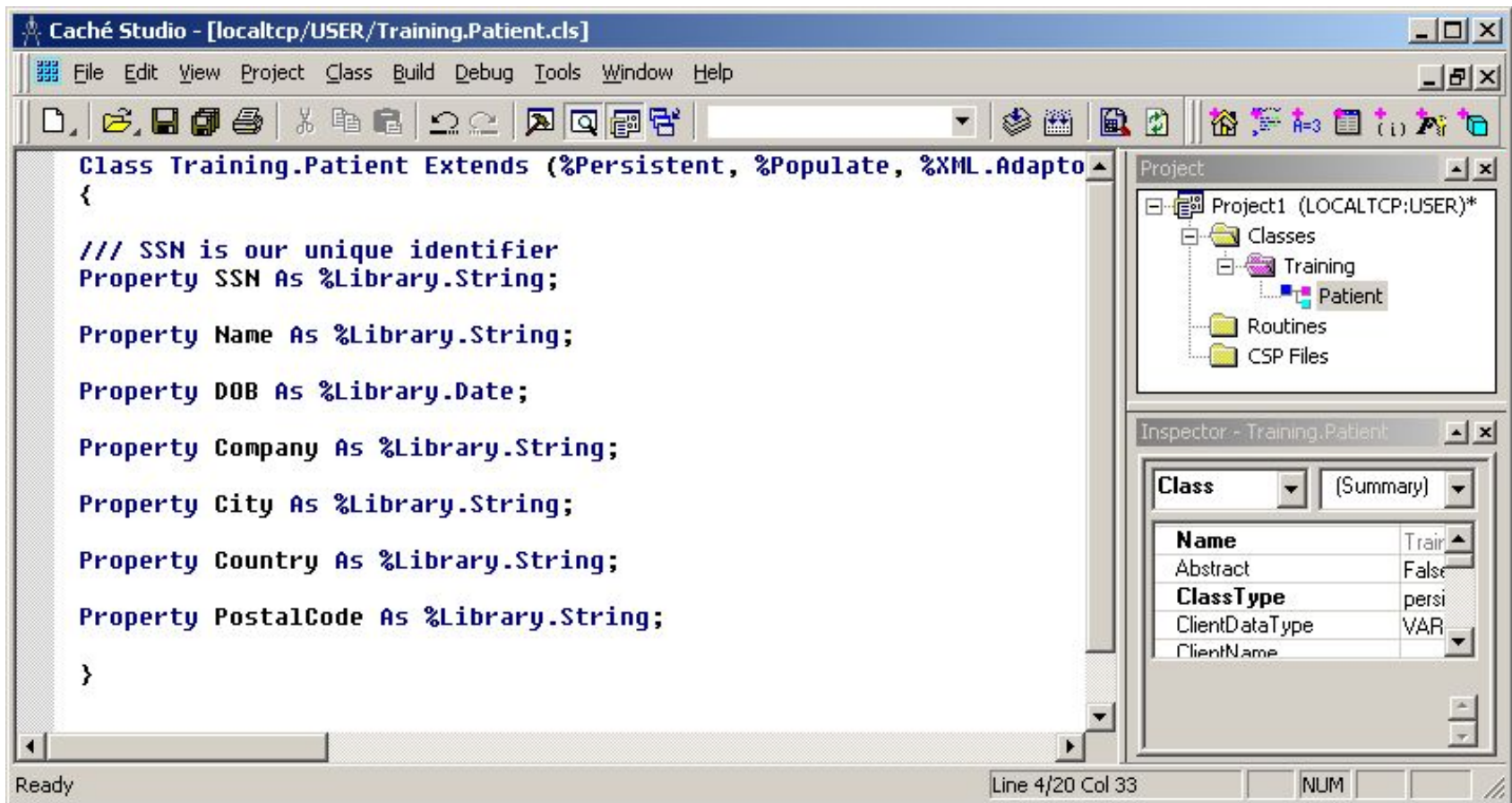
CacheSQLStorage Test

< Back Next > Finish Cancel Help

Добавляем свойства



Выбираем уникальный идентификатор



Caché Studio - [localhost/USER/Training.Patient.cls]

File Edit View Project Class Build Debug Tools Window Help

```
Class Training.Patient Extends (%Persistent, %Populate, %XML.Adaptor)
{
    /// SSN is our unique identifier
    Property SSN As %Library.String;

    Property Name As %Library.String;

    Property DOB As %Library.Date;

    Property Company As %Library.String;

    Property City As %Library.String;

    Property Country As %Library.String;

    Property PostalCode As %Library.String;
}
```

Project

- Project1 (LOCALTCP:USER)*
 - Classes
 - Training
 - Patient
 - Routines
 - CSP Files

Inspector - Training.Patient

Class (Summary)

Name	Train
Abstract	False
ClassType	persi
ClientDataType	VAR
ClientName	

Ready Line 4/20 Col 33 NUM

- Базируется на одном поле: SSN

Определяем ID / Primary Key индекс

New Index Wizard

Index Type

This index is:

☒ Normal: Used for maintaining an index on one or more properties

☒ This is a Unique Index

☒ This is the IDKEY for this class

☒ This is the SQL Primary key for this class

☐ Extent: Used for maintaining an index of all objects of this class within an extent.

This index is implemented as:

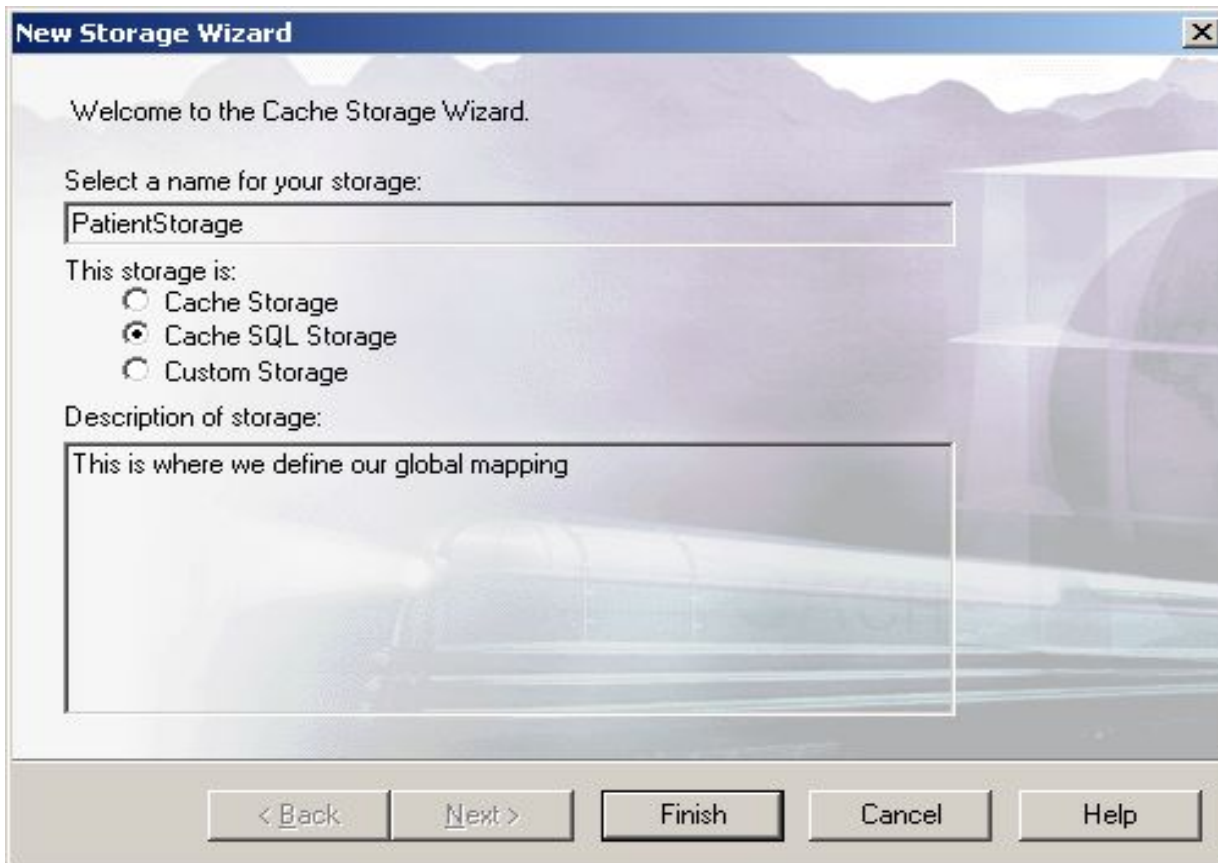
☒ a Standard Index.

☐ a Bitmap Index.

< Back Next > Finish Cancel Help

- Основан на свойстве SSN
- Не изменяйте collation индекса

Создаем Storage



The image shows a 'New Storage Wizard' dialog box with a blue title bar and a close button. The background features a faint image of a modern building and a train. The text inside the dialog is as follows:

Welcome to the Cache Storage Wizard.

Select a name for your storage:

PatientStorage

This storage is:

- ☐ Cache Storage
- ☒ Cache SQL Storage
- ☐ Custom Storage

Description of storage:

This is where we define our global mapping

At the bottom, there are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Создаем карту данных

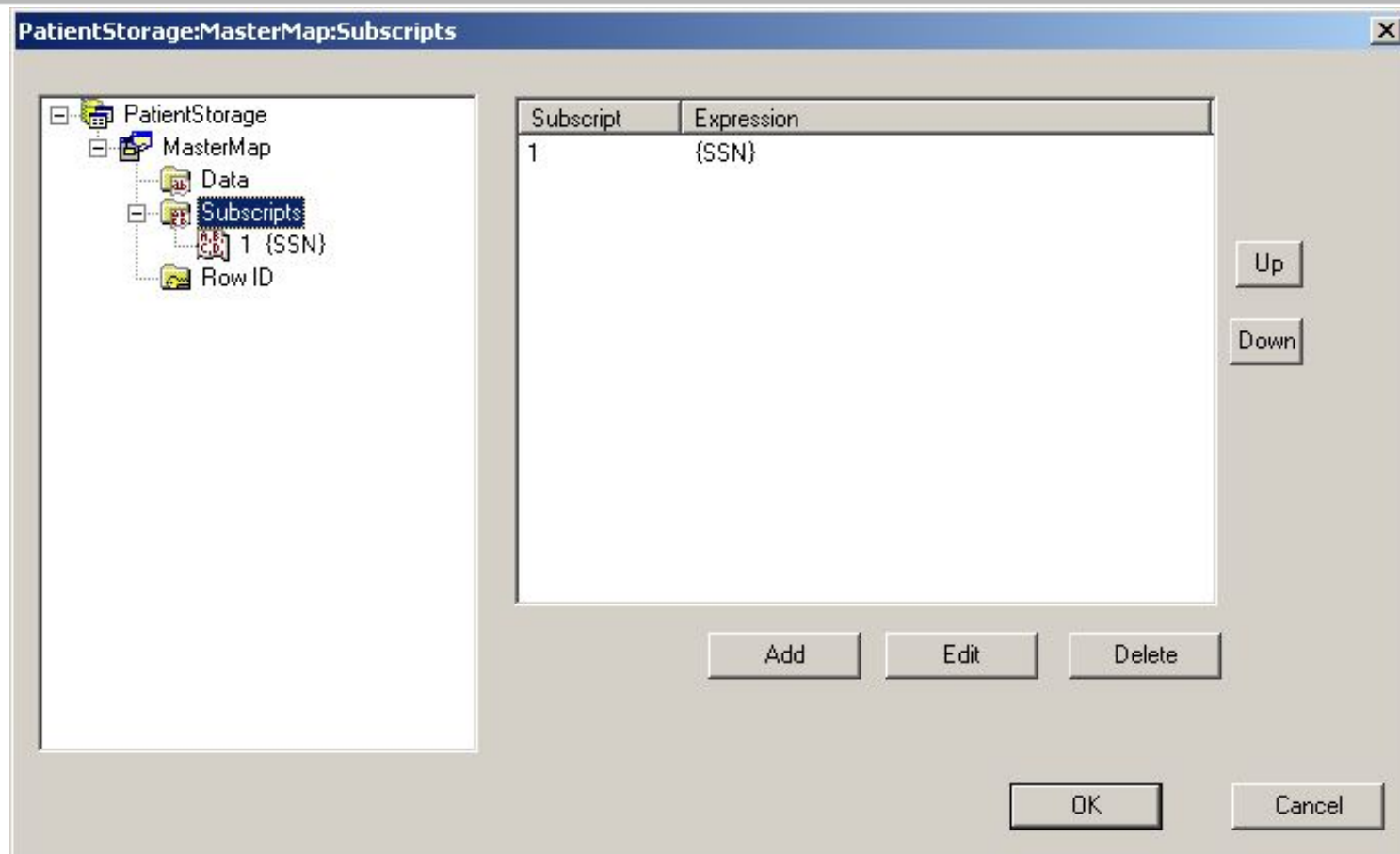
The screenshot shows a dialog box titled "PatientStorage:MasterMap". On the left is a tree view with "PatientStorage" and "MasterMap". The main area contains the following fields:

Map Name: <input type="text" value="MasterMap"/>	Map Type: <input type="text" value="data"/>
Global Name: <input type="text" value="^P"/>	Node Structure: <input type="text" value="\$Piece"/>
Population Type: <input type="text"/>	
Population %: <input type="text"/>	Condition: <input type="text"/>
Conditional Fields: <input type="text"/>	Conditional with hostvars: <input type="text"/>
Row reference: <input type="text"/>	

At the bottom right are "OK" and "Cancel" buttons.

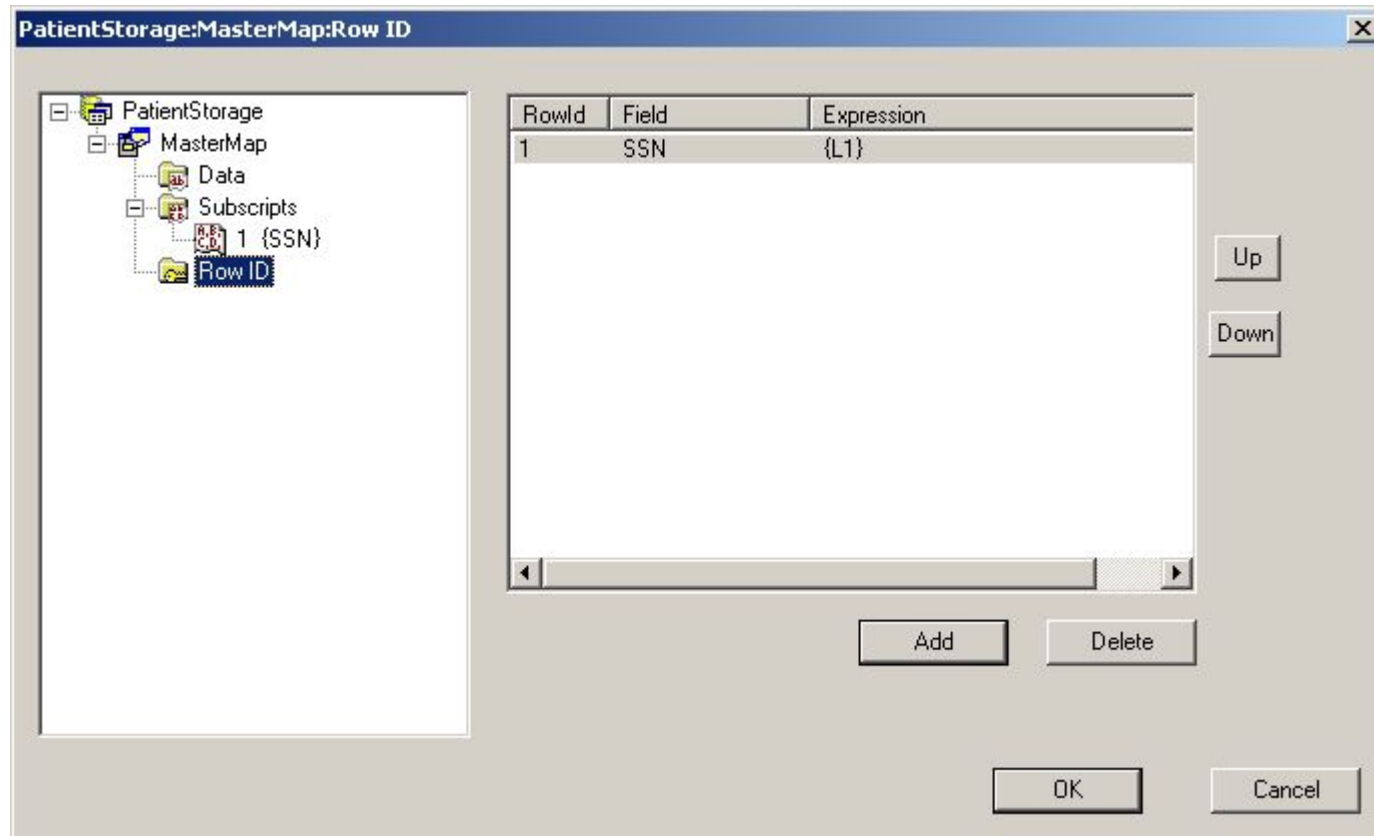
- Имя карты не может содержать символ «пробел»

Определяем индексы глобала



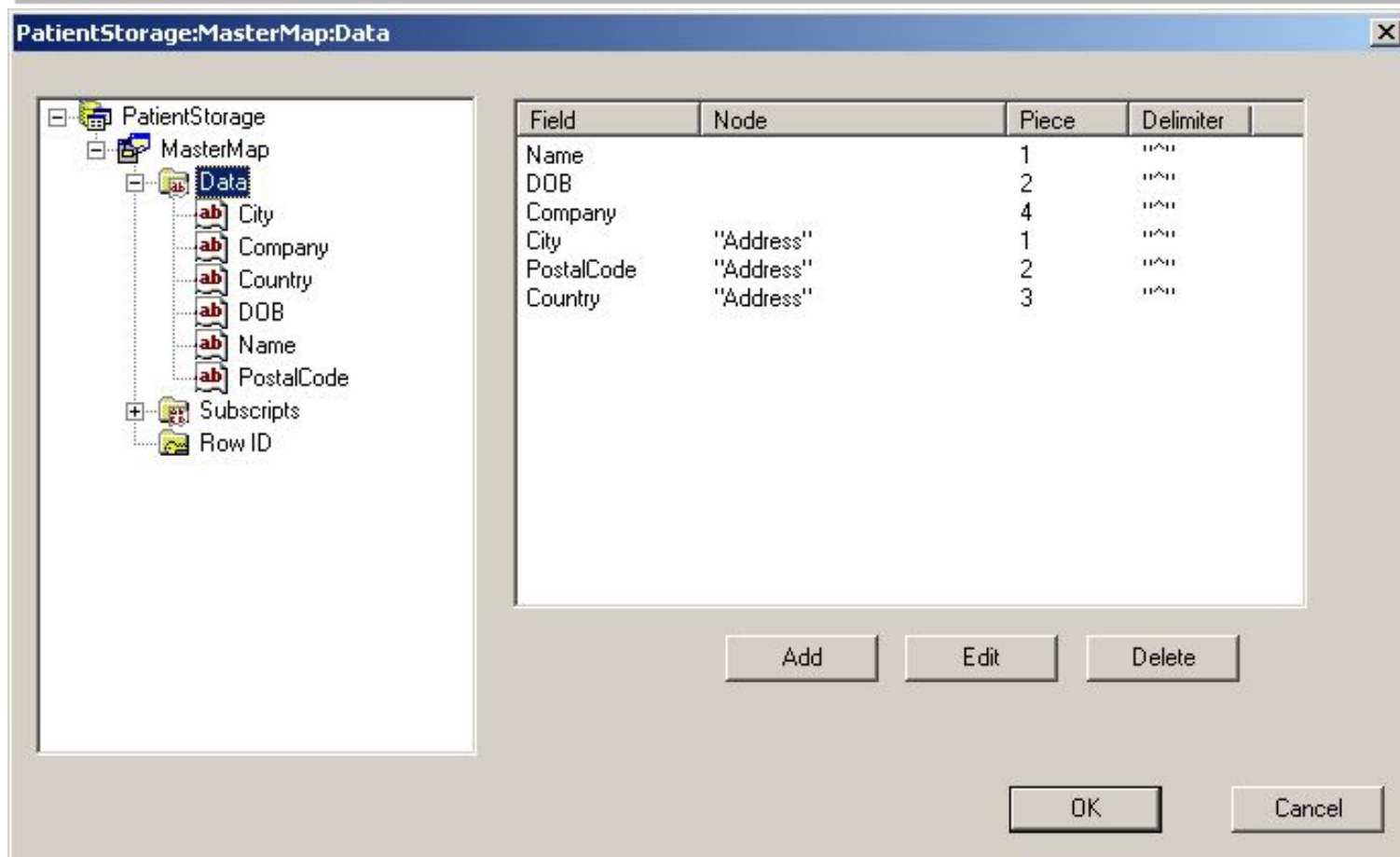
- Первый уровень индекса глобала - SSN

Определяем Row ID



- Первый Row ID 1 основан на SSN, которое хранится в первом уровне индекса глобала

Определяем свойства



- Введите разделитель и дополнительную информацию

Создаем карту индексов

PatientStorage:Map1

Tree View:

- PatientStorage
 - MasterMap
 - NameIndex

Map Name: NameIndex

Map Type: index

Global Name: ^PI

Node Structure: \$Piece

Population Type: full

Population %:

Condition:

Conditional Fields:

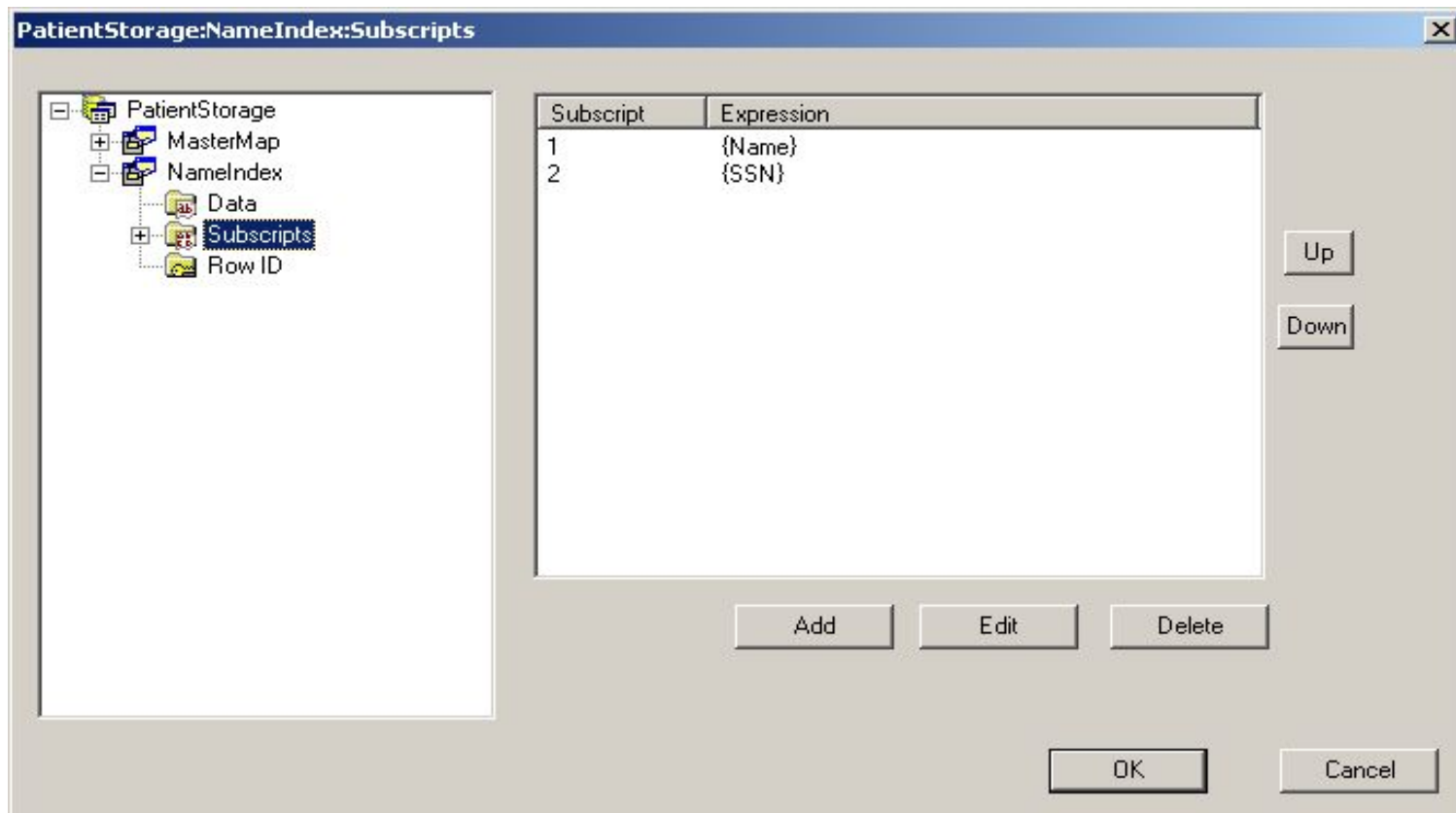
Conditional with hostvars:

Row reference:

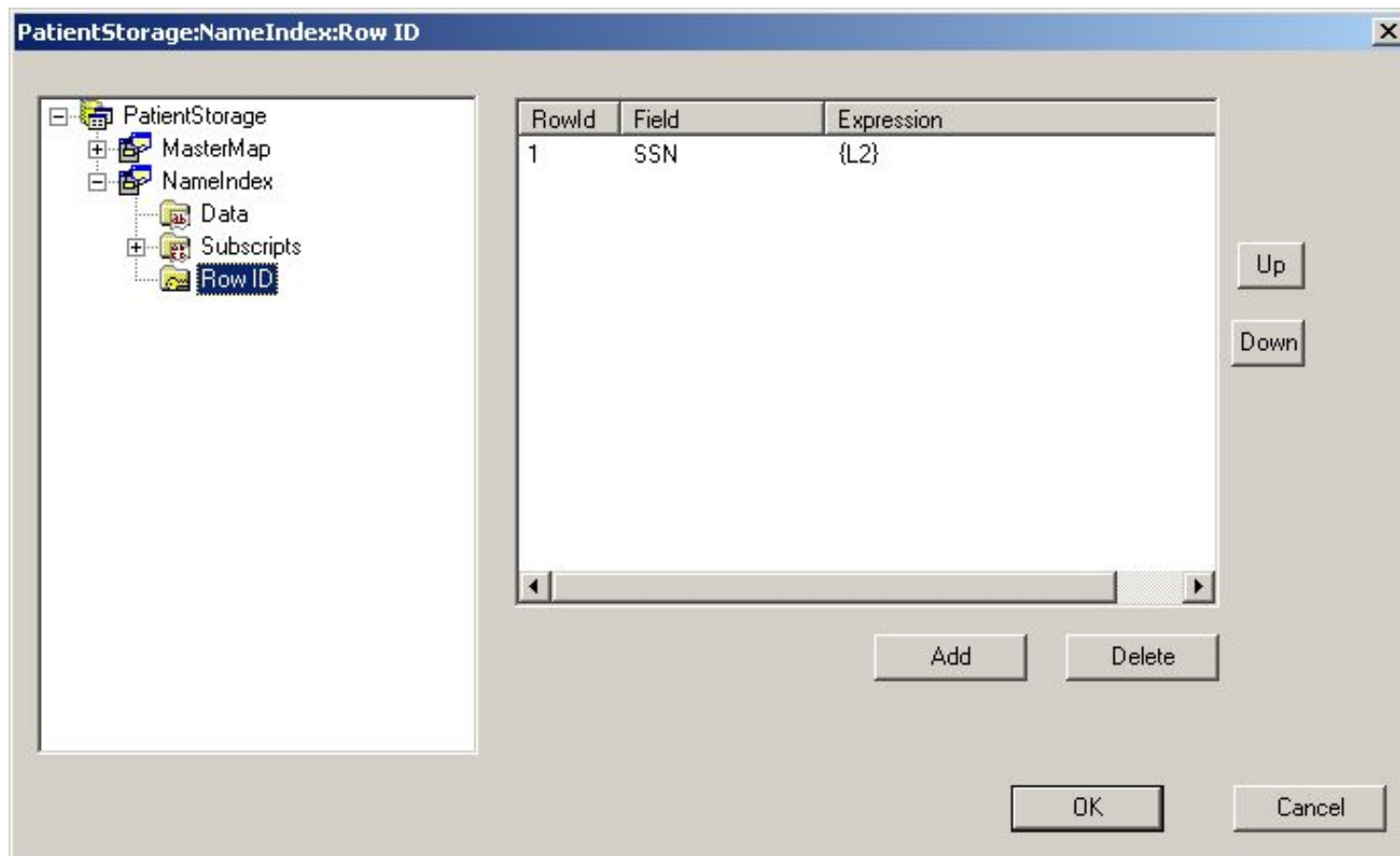
OK Cancel

- Выберите тип заполнения 'full'

Определяем индексы глобала индексов

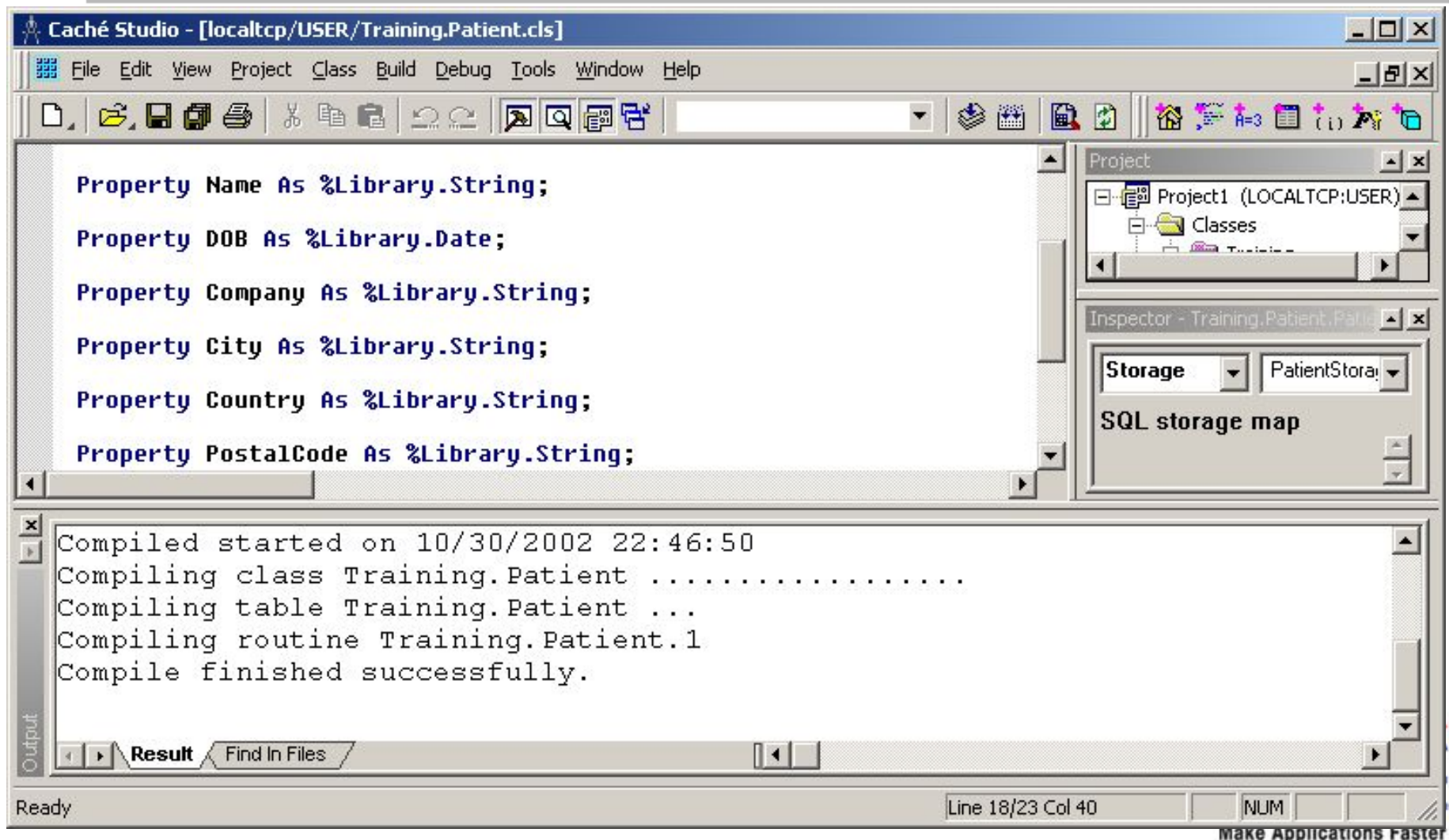


Определяем Row ID индекса

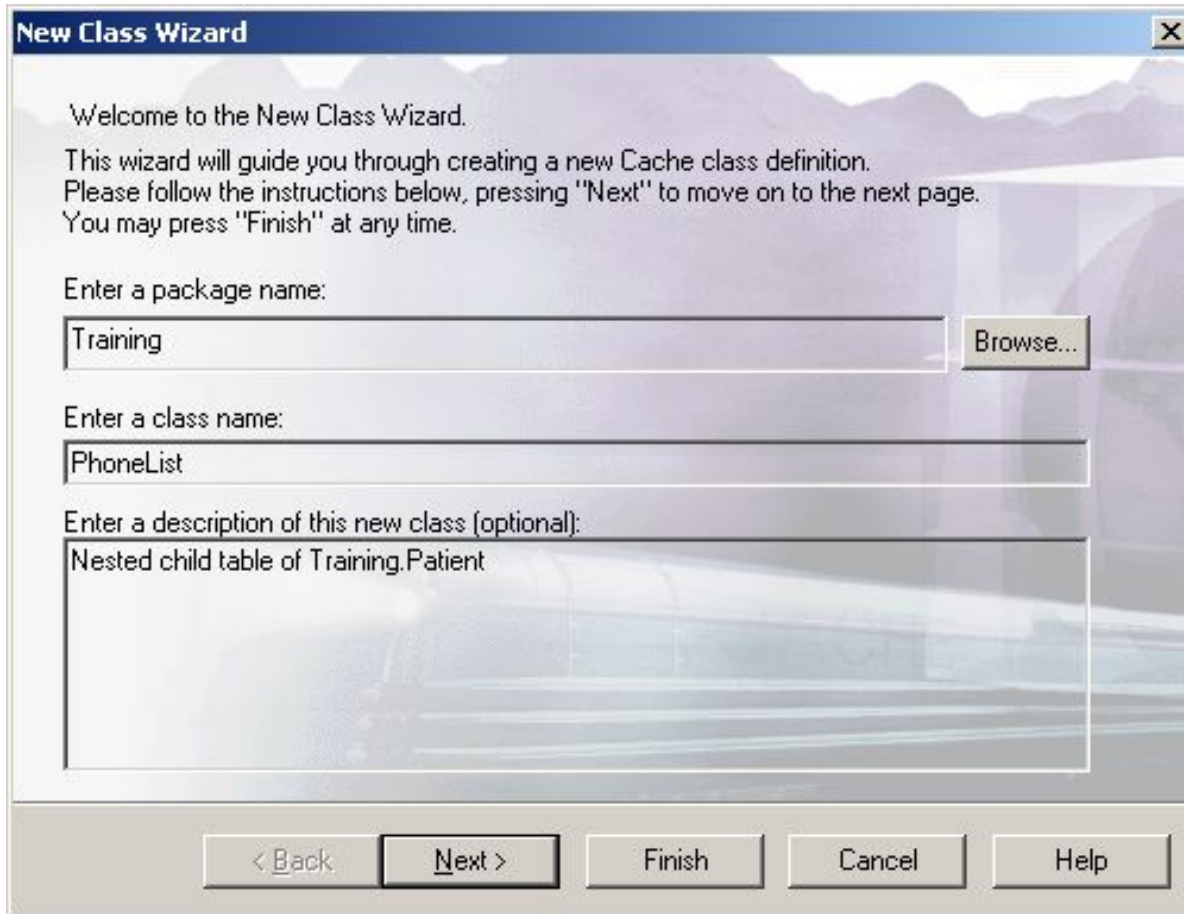


- Первый Row ID основан на SSN, который хранится в уровне 2 индекса глобала

Сохраняем и компилируем класс



Создаем дочернюю таблицу PhoneList



New Class Wizard

Welcome to the New Class Wizard.
This wizard will guide you through creating a new Cache class definition.
Please follow the instructions below, pressing "Next" to move on to the next page.
You may press "Finish" at any time.

Enter a package name:

Training Browse...

Enter a class name:

PhoneList

Enter a description of this new class (optional):

Nested child table of Training.Patient

< Back Next > Finish Cancel Help

- Этот класс тоже Persistent

Создаем отношение Parent-Child

Relationship Wizard

Relationship Characteristics

This property is one side of a relationship between this object and one or more other objects.

This relationship property refers to:

- ☐ One: one other object
- ☐ Many: many other objects
- ☒ Parent: this object's parent
- ☐ Children: this object's children

This relationship property references objects of the following type:

Training.Patient

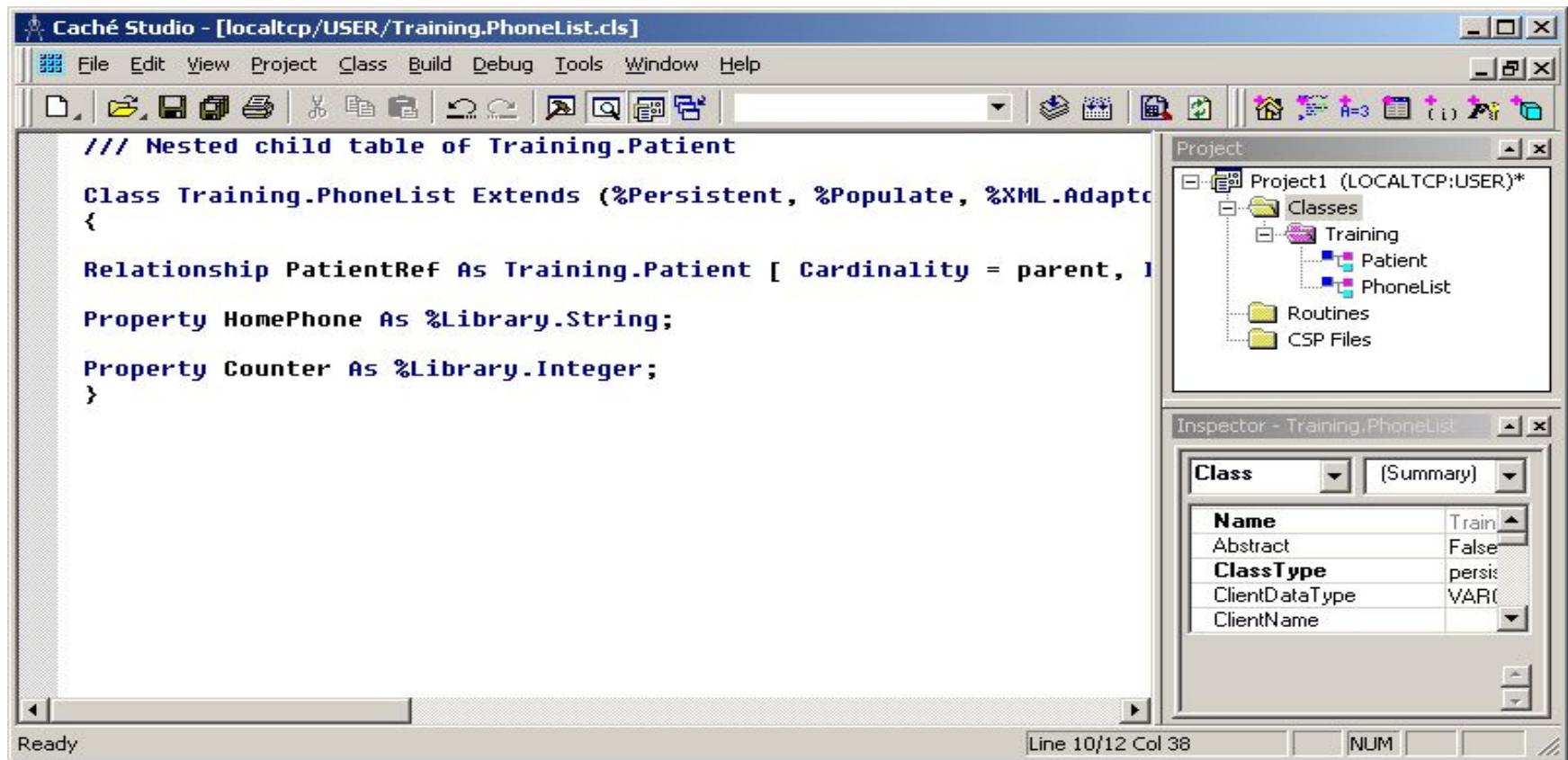
The name of the corresponding property in the referenced class is:

PhoneNumbers

< Back Next > Finish Cancel Help

- Отношения (Relationship) – специальный класс свойств
- Кроме определения свойства в этом классе (PatientRef), Вы должны определить другую сторону отношения (PhoneNumbers)

Добавляем остальные свойства



The screenshot shows the Caché Studio IDE with the file `Training.PhoneList.cls` open. The code defines a class `Training.PhoneList` that extends `%Persistent`, `%Populate`, and `%XML.Adaptor`. It includes a relationship `PatientRef` to `Training.Patient` with a cardinality of `parent, 1`. Two properties are defined: `HomePhone` of type `%Library.String` and `Counter` of type `%Library.Integer`.

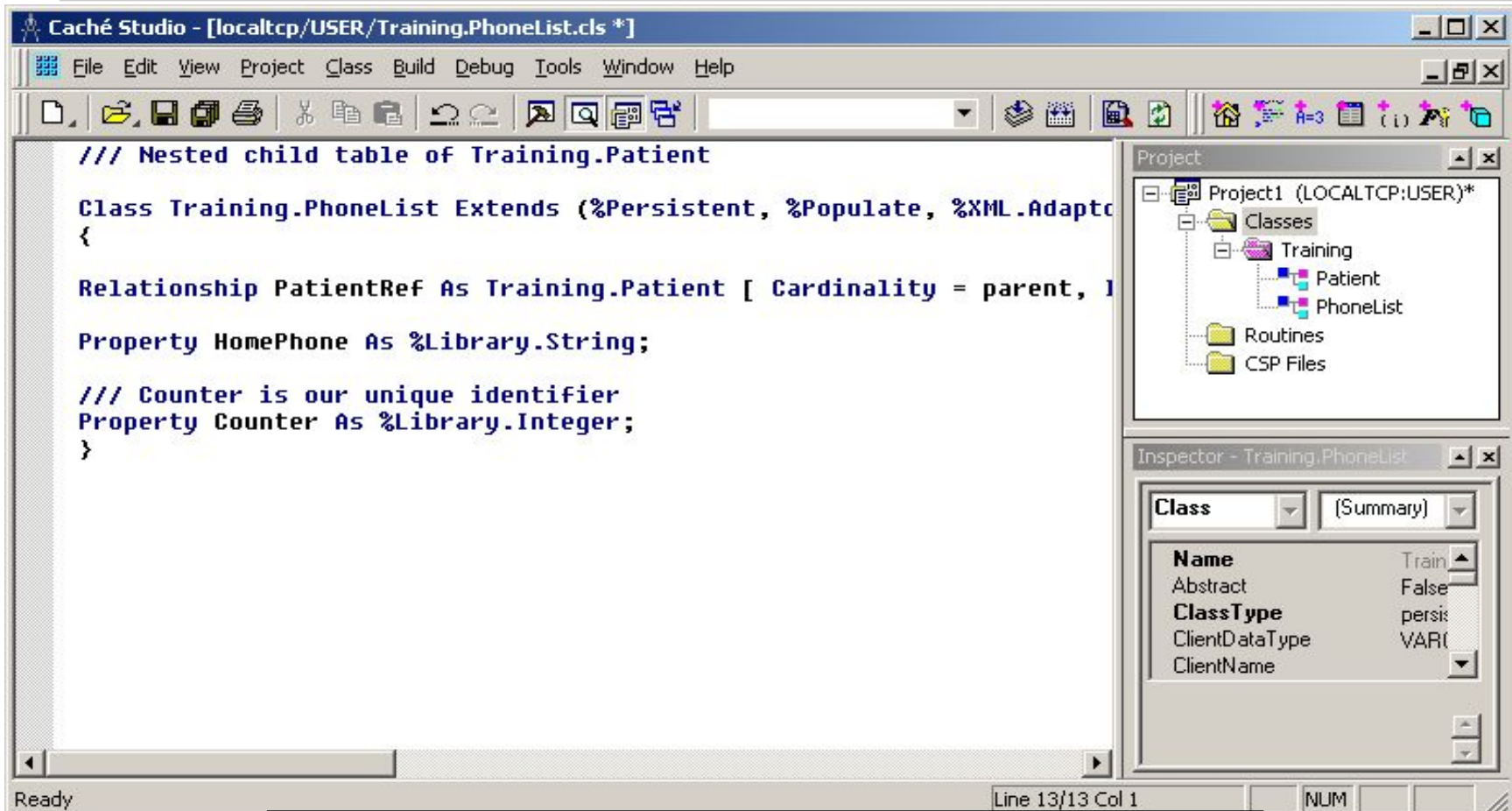
```
/// Nested child table of Training.Patient  
Class Training.PhoneList Extends (%Persistent, %Populate, %XML.Adaptor)  
{  
    Relationship PatientRef As Training.Patient [ Cardinality = parent, 1  
    Property HomePhone As %Library.String;  
    Property Counter As %Library.Integer;  
}
```

The Project Explorer on the right shows the project structure with `Classes` containing `Patient` and `PhoneList`. The Inspector on the right shows the properties of the `Training.PhoneList` class:

Name	Value
Abstract	False
ClassType	persis
ClientDataType	VARC
ClientName	

- Кроме свойства, Вы должны определить свойство для представления позиции во встроенной разделенной (Counter)

Выбираем уникальный идентификатор



The screenshot shows the Caché Studio IDE with the file `Training.PhoneList.cls` open. The code defines a class `Training.PhoneList` that extends `(%Persistent, %Populate, %XML.Adapt)`. It includes a relationship `PatientRef` to `Training.Patient` with a cardinality of `parent, 1`, and a property `HomePhone` of type `%Library.String`. A comment indicates that the `Counter` property is used as a unique identifier.

```
/// Nested child table of Training.Patient  
  
Class Training.PhoneList Extends (%Persistent, %Populate, %XML.Adapt)  
{  
  
    Relationship PatientRef As Training.Patient [ Cardinality = parent, 1  
  
    Property HomePhone As %Library.String;  
  
    /// Counter is our unique identifier  
    Property Counter As %Library.Integer;  
}
```

The Project Explorer on the right shows the project structure:

- Project1 (LOCALTCP:USER)*
 - Classes
 - Training
 - Patient
 - PhoneList
 - Routines
 - CSP Files

The Inspector panel shows the properties of the `Training.PhoneList` class:

Class	
Name	Train
Abstract	False
Class Type	persi
ClientData Type	VAR
ClientName	

- Наш идентификатор будет строиться на поле Counter

Определяем индекс ID / Primary Key

New Index Wizard

Index Type

This index is:

- ☒ Normal: Used for maintaining an index on one or more properties
 - ☒ This is a Unique Index
 - ☒ This is the IDKEY for this class
 - ☒ This is the SQL Primary key for this class
- ☐ Extent: Used for maintaining an index of all objects of this class within an extent.

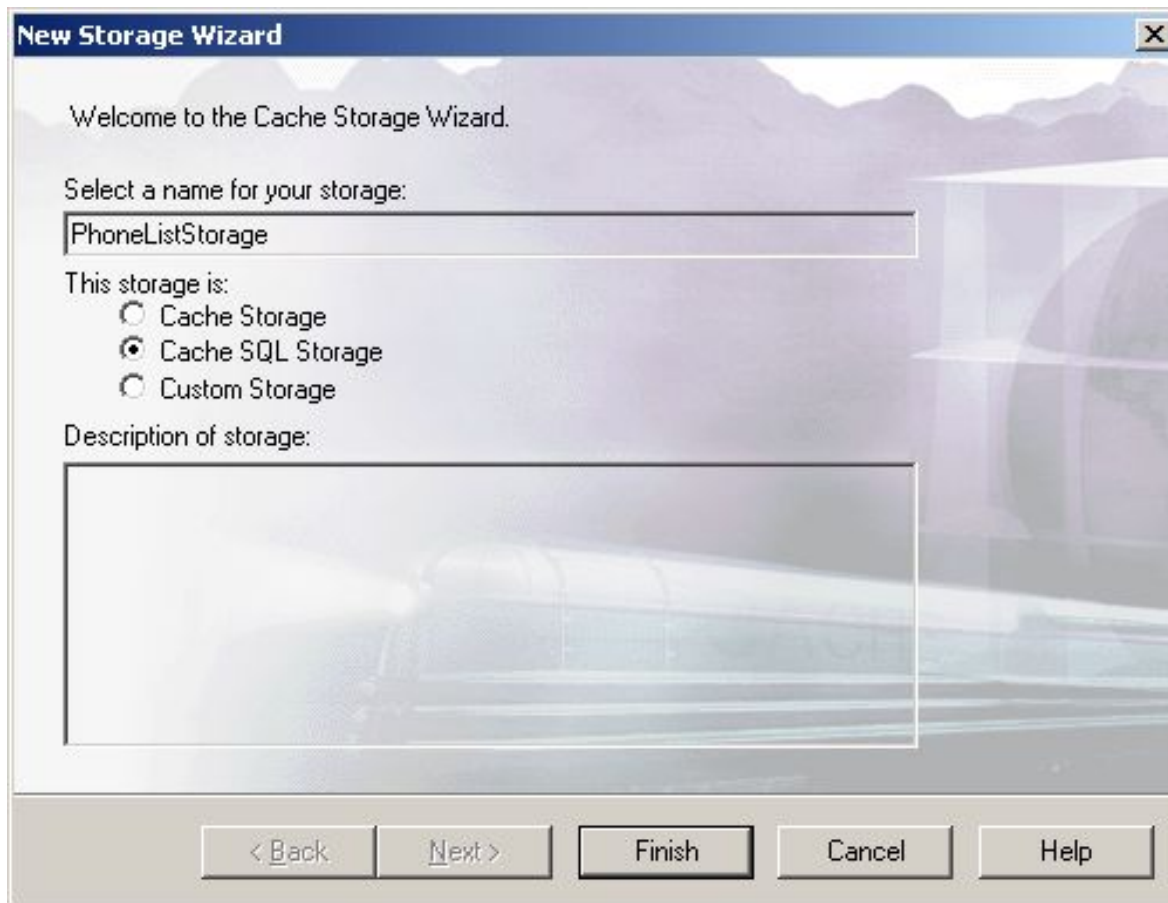
This index is implemented as:

- ☒ a Standard Index.
- ☐ a Bitmap Index.

< Back Next > Finish Cancel Help

- Задайте индекс по свойству Counter
- Не модифицируйте collation
- Свойство PatientRef неявно часть IDKey / Primary Key

Создаем Storage



The image shows a 'New Storage Wizard' dialog box with a blue title bar and a close button. The background of the dialog features a faint, stylized image of a modern building and mountains. The text inside the dialog is as follows:

Welcome to the Cache Storage Wizard.

Select a name for your storage:

PhoneListStorage

This storage is:

- ☐ Cache Storage
- ☒ Cache SQL Storage
- ☐ Custom Storage

Description of storage:

Below the description label is a large, empty rectangular text area.

At the bottom of the dialog, there are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Создаем карту данных

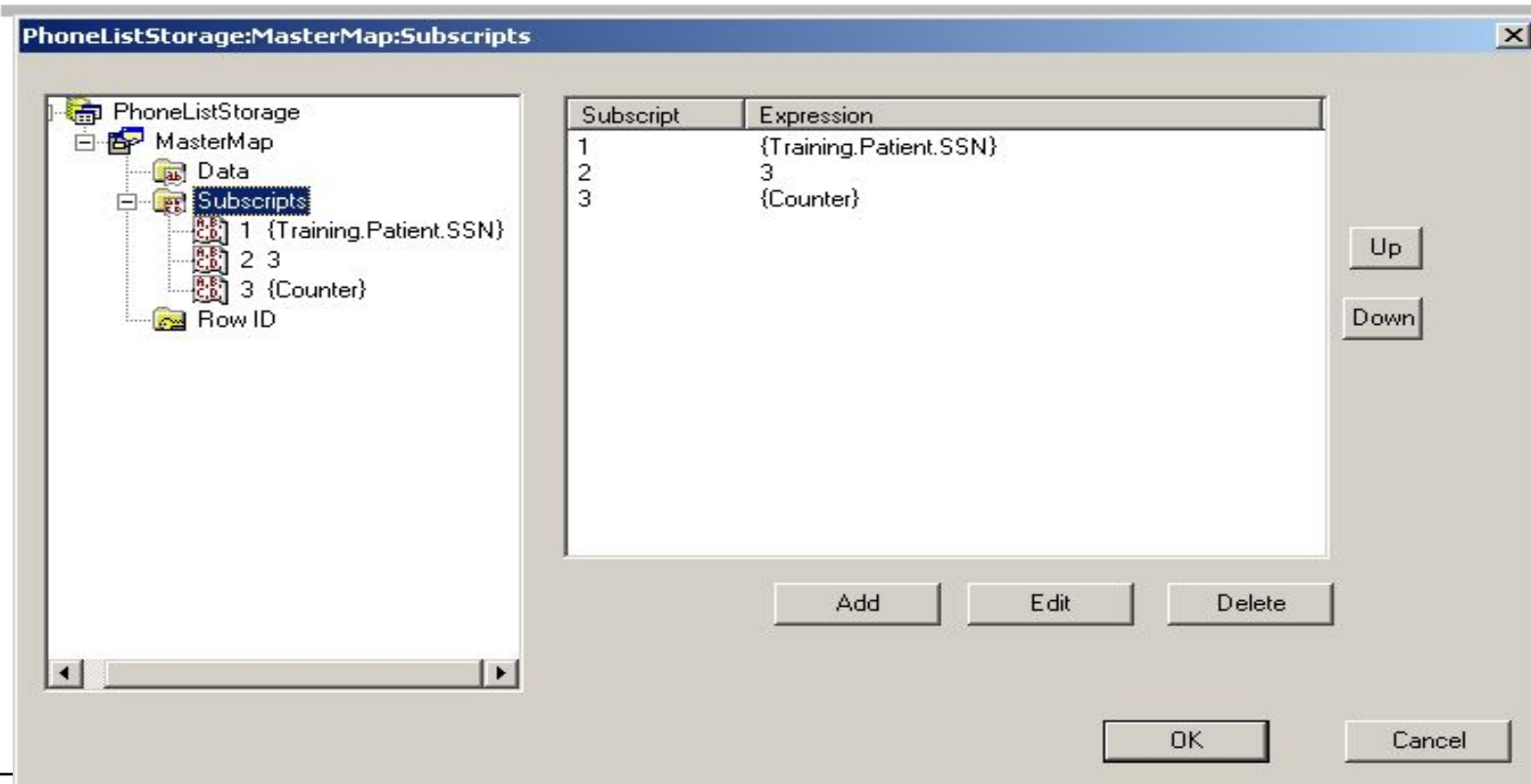
The screenshot shows a dialog box titled "PatientStorage:MasterMap". On the left is a tree view with "PatientStorage" and "MasterMap". The main area contains the following fields:

- Map Name:
- Map Type:
- Global Name:
- Node Structure:
- Population Type:
- Population %:
- Condition:
- Conditional Fields:
- Conditional with hostvars:
- Row reference:

At the bottom right are "OK" and "Cancel" buttons.

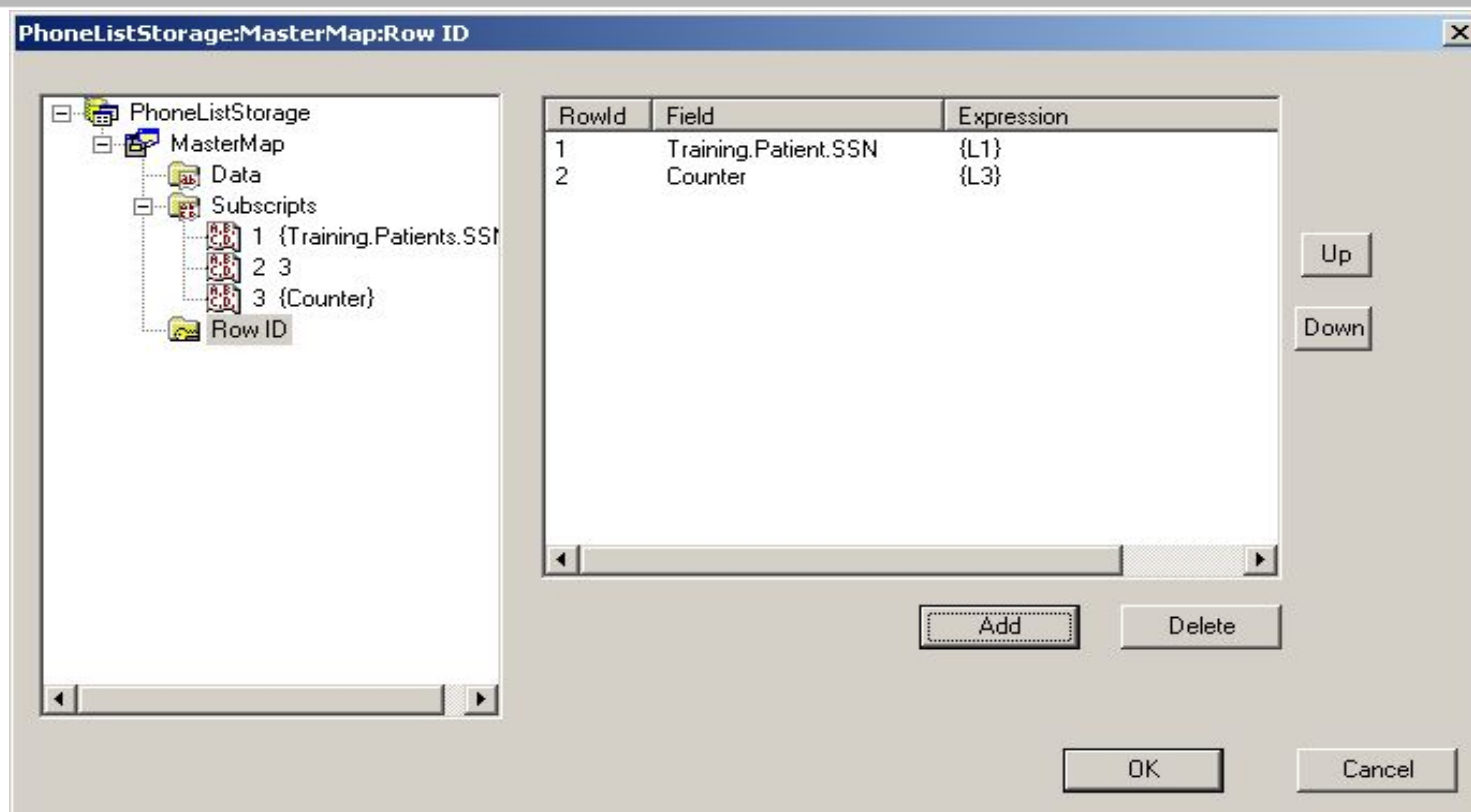
- Имя карты не может содержать символ «пробел»

Определяем индексы глобала



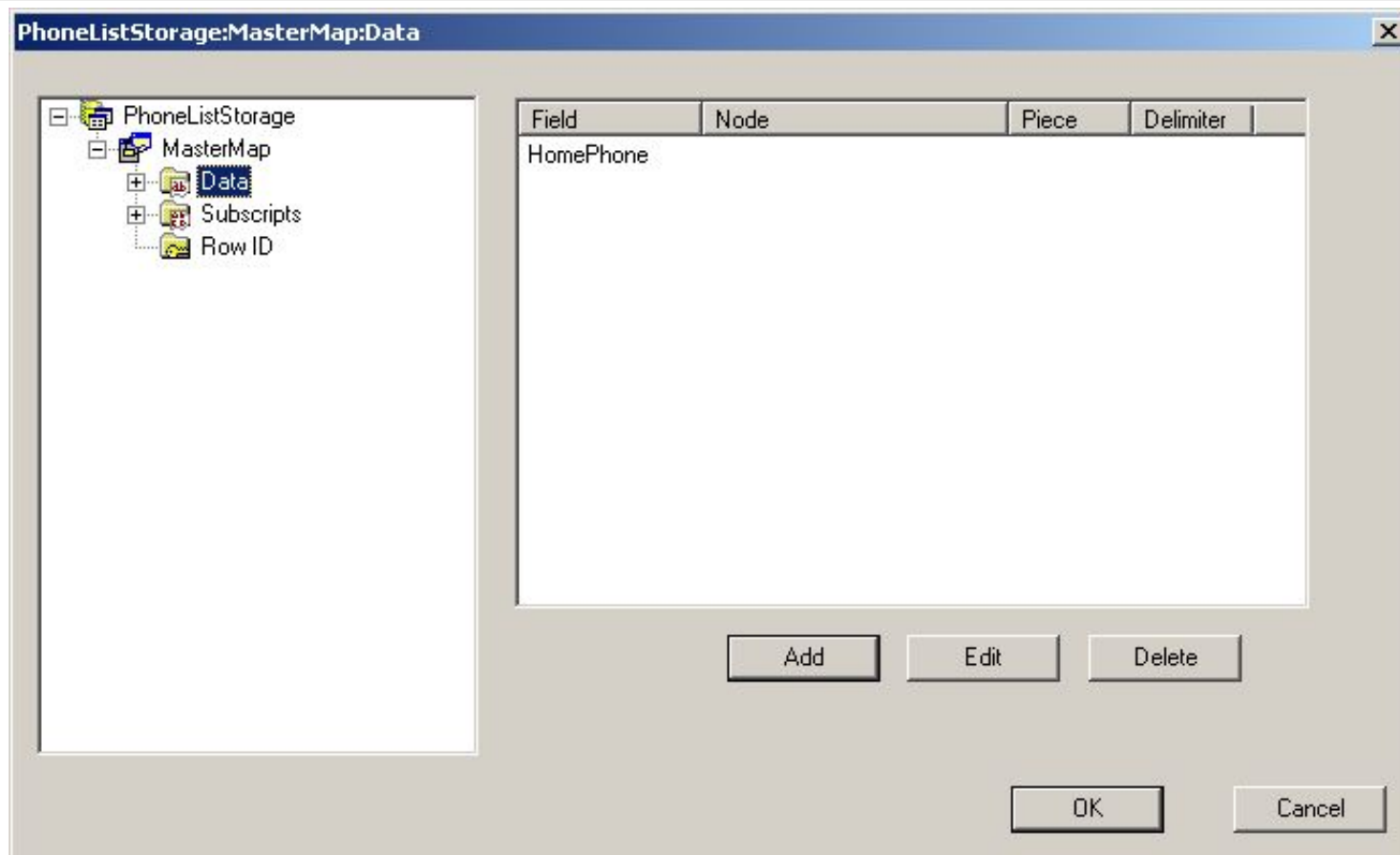
- Первый уровень индекса глобала основан на Training.Patient.SSN
- Второй уровень индекса глобала основан на “^” разделителе, используя третью позицию
- Третий уровень индекса глобала основан на “~” разделителе, используя свойство Counter для позиции

Определяем Row ID



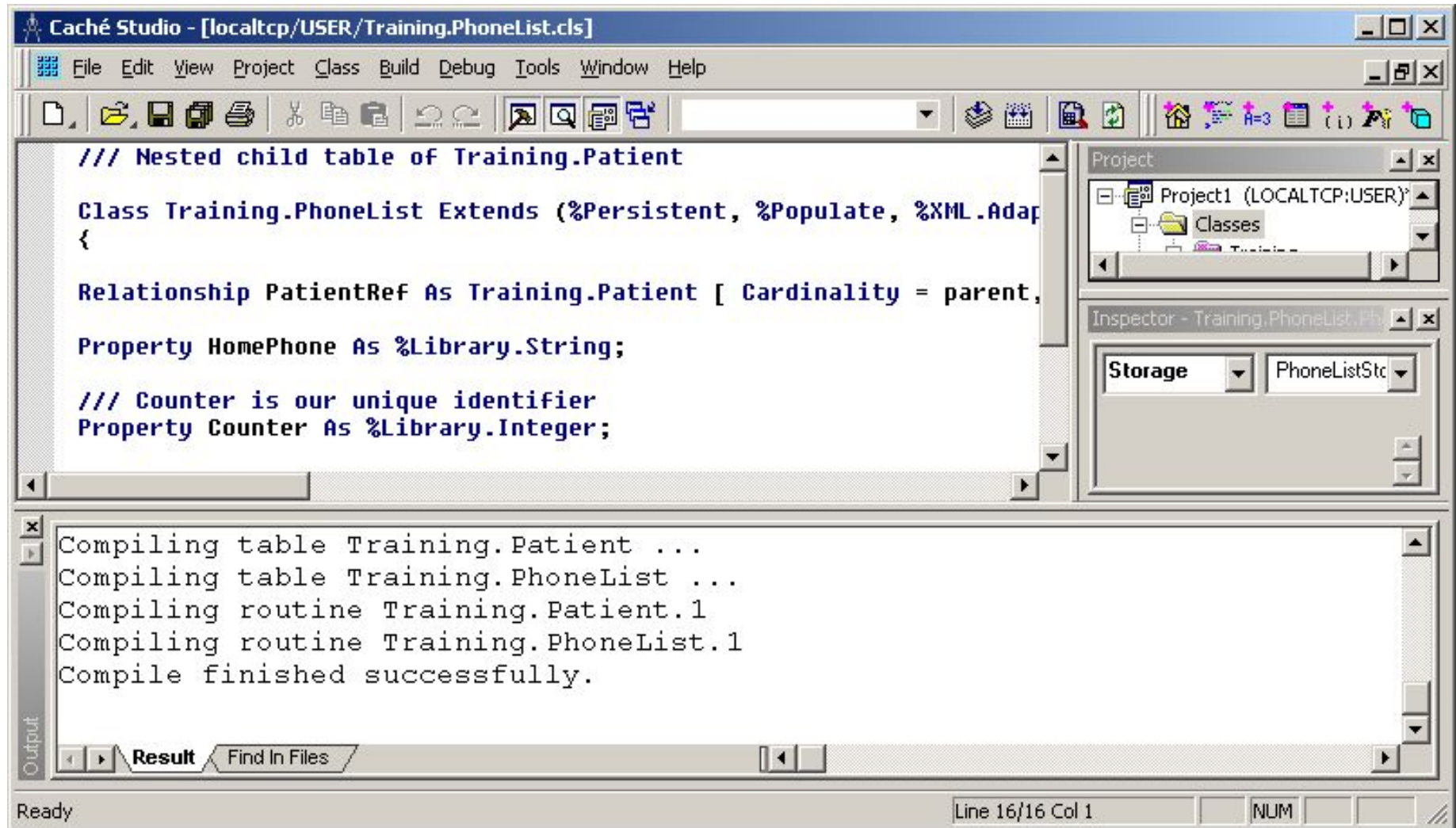
- Row ID 1 основан на Training.Patient.SSN, которое хранится на первом уровне индекса глобала
- Row ID 2 основан на Counter, которое хранится на первом уровне индекса глобала

Определяем свойства



- Внесите только свойство HomePhone

Сохраняем и компилируем класс



Создаем дочернюю таблицу Visit

New Class Wizard

Welcome to the New Class Wizard.
This wizard will guide you through creating a new Cache class definition.
Please follow the instructions below, pressing "Next" to move on to the next page.
You may press "Finish" at any time.

Enter a package name:

Training Browse...

Enter a class name:

Visit

Enter a description of this new class (optional):

Sub-node child table of Training.Patient

< Back Next > Finish Cancel Help

- Этот класс тоже Persistent

Создаем отношение Parent-Child

Relationship Wizard

Relationship Characteristics

This property is one side of a relationship between this object and one or more other objects.
This relationship property refers to:

☐ One: one other object
☐ Many: many other objects
☒ Parent: this object's parent
☐ Children: this object's children

This relationship property references objects of the following type:

Training.Patient

The name of the corresponding property in the referenced class is:

Visits

- Кроме определения свойства в этом классе (PatientRef), Вы должны определить другую сторону отношения (Visits)

Добавляем остальные свойства

The screenshot shows the Caché Studio IDE with the following components:

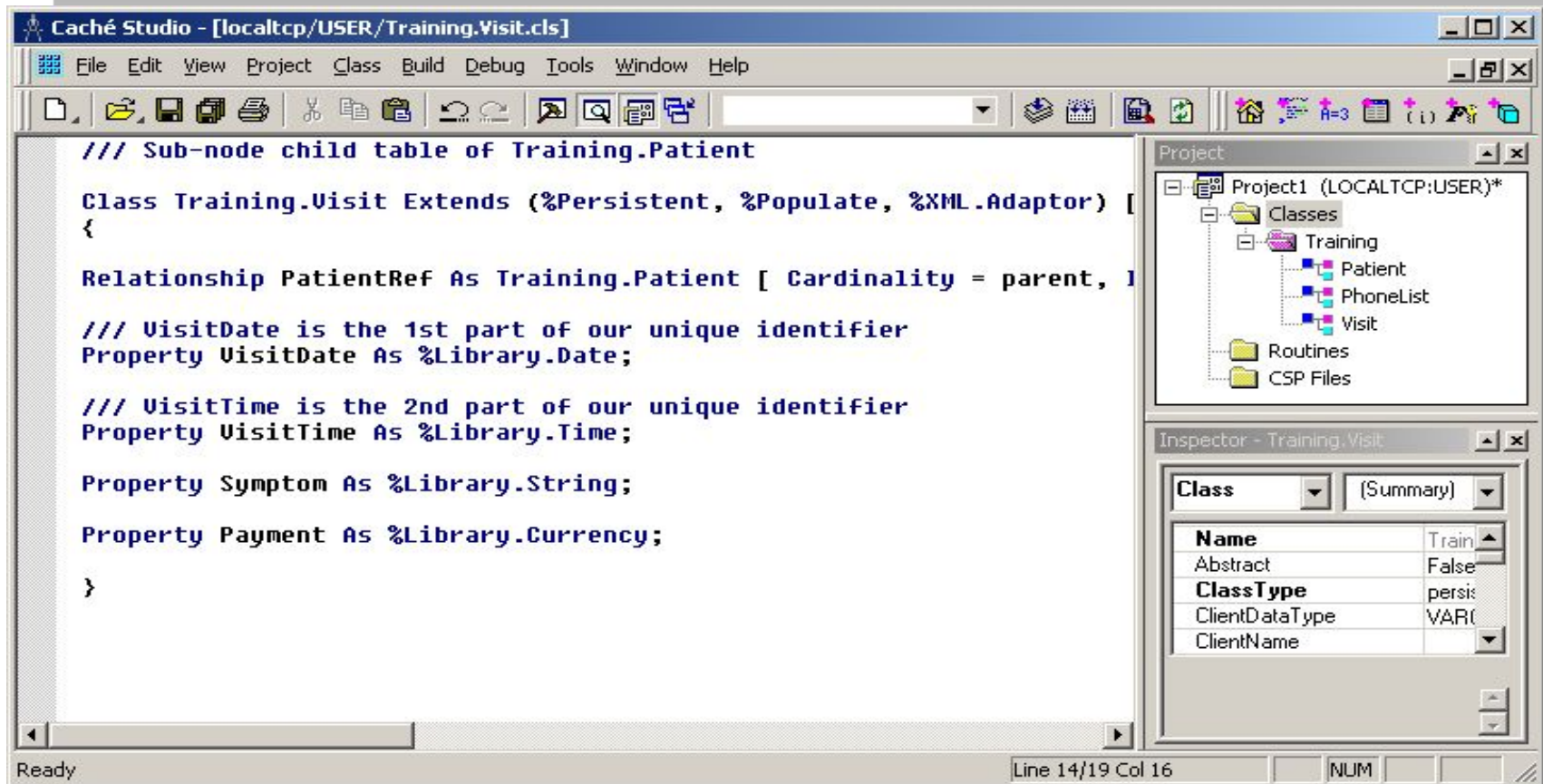
- Title Bar:** Caché Studio - [localtcp/USER/Training.Visit.cls]
- Menu Bar:** File, Edit, View, Project, Class, Build, Debug, Tools, Window, Help
- Toolbar:** Standard IDE icons for file operations, editing, and debugging.
- Source Code Editor:** Contains the following code:

```
/// Sub-node child table of Training.Patient  
  
Class Training.Visit Extends (%Persistent, %Populate, %XML.Adaptor) [  
{  
  
    Relationship PatientRef As Training.Patient [ Cardinality = parent, 1  
  
    Property VisitDate As %Library.Date;  
  
    Property VisitTime As %Library.Time;  
  
    Property Symptom As %Library.String;  
  
    Property Payment As %Library.Currency;  
  
}
```
- Project Pane:** Shows the project structure for Project1 (LOCALTCP:USER)*. The 'Classes' folder contains Training, which includes Patient, PhoneList, and Visit. Other folders are Routines and CSP Files.
- Inspector Pane:** Titled 'Inspector - Training.Visit.Payment', it shows the properties of the Payment property. The 'Property' dropdown is set to 'Payment'. The properties listed are:

Property	Value
XMLNAME	
XMLPROJECTION	
XSDTYPE	decin
Private	False
ReadOnly	False

The 'Parameter' section below shows 'Parameter'.
- Status Bar:** Displays 'Ready' on the left and 'Line 17/17 Col 1' on the right.

Выбираем уникальный идентификатор



The screenshot shows the Caché Studio IDE with the following components:

- Title Bar:** Caché Studio - [localtcp/USER/Training.Visit.cls]
- Menu Bar:** File, Edit, View, Project, Class, Build, Debug, Tools, Window, Help
- Toolbar:** Standard development tools like file operations, search, and execution.
- Main Editor:** Contains the source code for the `Training.Visit` class.

```
/// Sub-node child table of Training.Patient

Class Training.Visit Extends (%Persistent, %Populate, %XML.Adaptor) {
{

Relationship PatientRef As Training.Patient [ Cardinality = parent, 1

/// VisitDate is the 1st part of our unique identifier
Property VisitDate As %Library.Date;

/// VisitTime is the 2nd part of our unique identifier
Property VisitTime As %Library.Time;

Property Symptom As %Library.String;

Property Payment As %Library.Currency;

}
```
- Project Pane:** Shows the project structure for 'Project1 (LOCALTCP:USER)*'. It includes a 'Classes' folder containing 'Training', which in turn contains 'Patient', 'PhoneList', and 'Visit'. There are also 'Routines' and 'CSP Files' folders.
- Inspector Pane:** Titled 'Inspector - Training.Visit', it shows properties for the selected class. The 'Class' dropdown is set to 'Training.Visit' and the 'Summary' dropdown is set to '(Summary)'. The properties listed are:

Name	Value
Abstract	False
Class Type	persis
ClientData Type	VARC
ClientName	
- Status Bar:** Shows 'Ready' on the left and 'Line 14/19 Col 16' on the right.

- В этот раз идентификатор будет строиться по 2 полям: VisitDate и VisitTime

Определяем индекс ID / Primary Key

New Index Wizard

Index Type

This index is:

- ☒ Normal: Used for maintaining an index on one or more properties
 - ☒ This is a Unique Index
 - ☒ This is the IDKEY for this class
 - ☒ This is the SQL Primary key for this class
- ☐ Extent: Used for maintaining an index of all objects of this class within an extent.

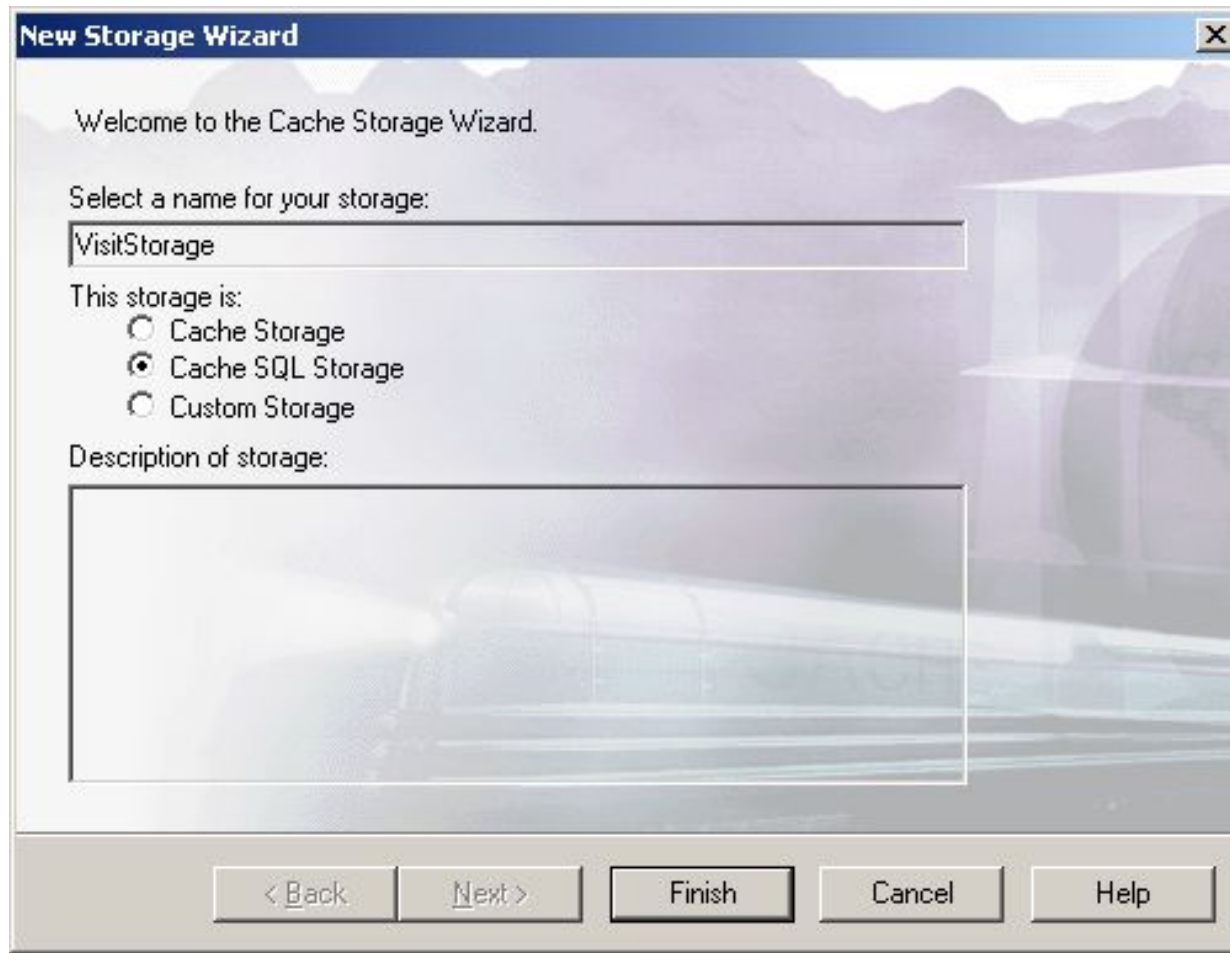
This index is implemented as:

- ☒ a Standard Index.
- ☐ a Bitmap Index.

< Back Next > Finish Cancel Help

- Постройте индекс по свойствам VisitDate и VisitTime
- Не изменяйте collation
- Свойство PatientRef неявно часть IDKey / Primary Key

Создаем Storage



The image shows a 'New Storage Wizard' dialog box with a blue title bar and a close button. The background features a faint image of a futuristic building. The text inside the dialog is as follows:

Welcome to the Cache Storage Wizard.

Select a name for your storage:

VisitStorage

This storage is:

- ☐ Cache Storage
- ☒ Cache SQL Storage
- ☐ Custom Storage

Description of storage:

Below the description label is a large, empty rectangular text area.

At the bottom of the dialog are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Создаем карту данных

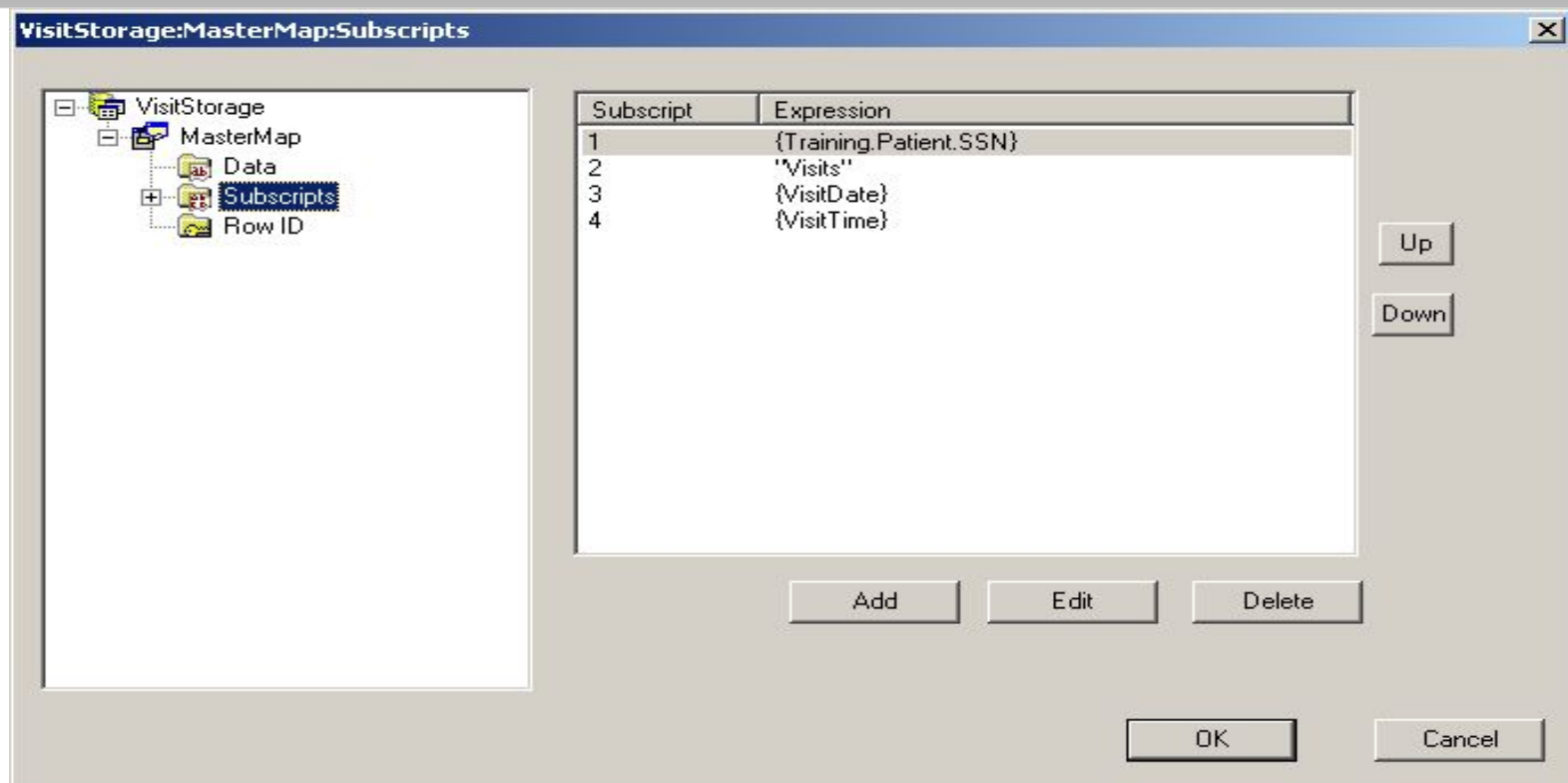
The screenshot shows a dialog box titled "PatientStorage:MasterMap". On the left is a tree view with "PatientStorage" and "MasterMap". The main area contains the following fields:

- Map Name:
- Map Type:
- Global Name:
- Node Structure:
- Population Type:
- Population %:
- Condition:
- Conditional Fields:
- Conditional with hostvars:
- Row reference:

At the bottom right are "OK" and "Cancel" buttons.

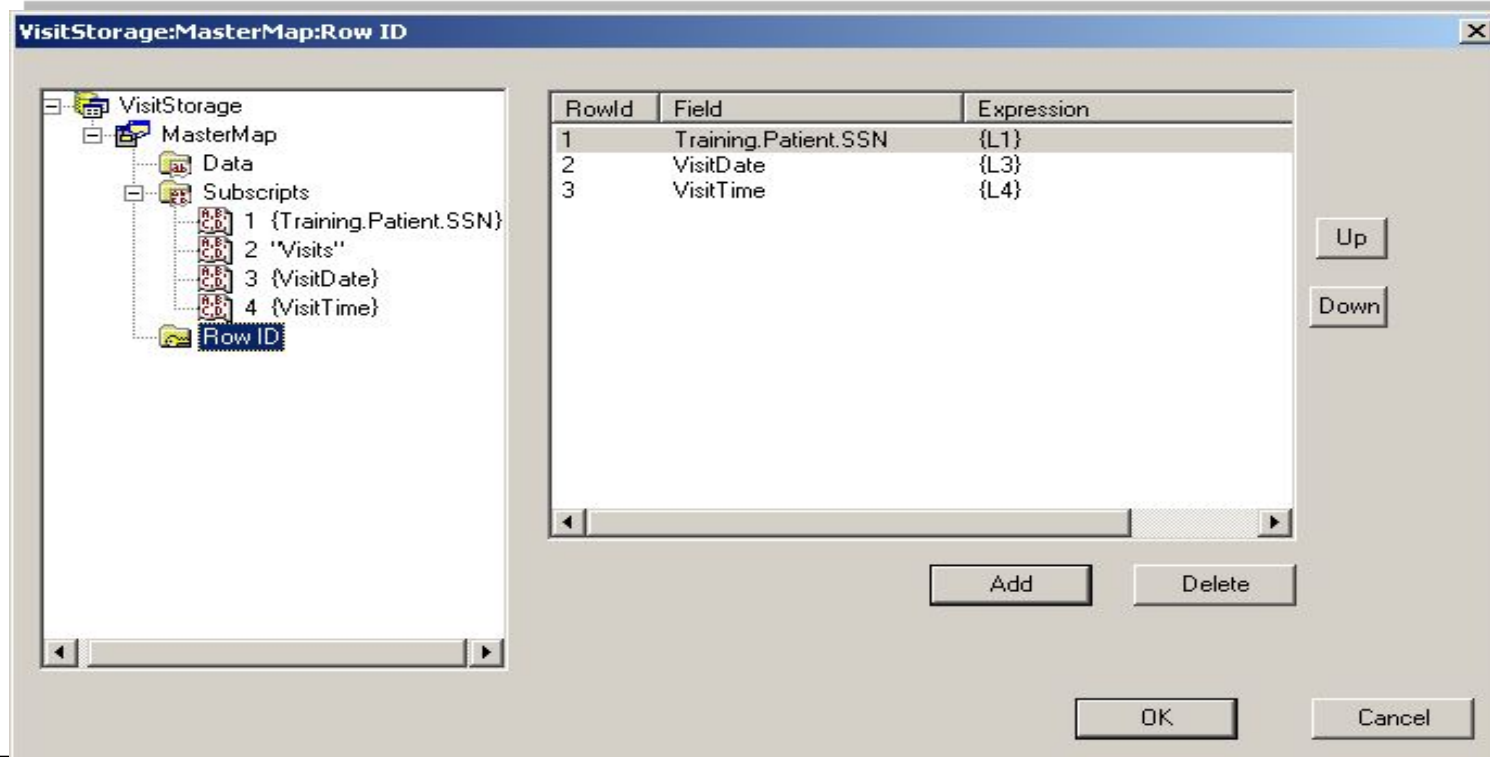
- Имя карты не может содержать символ «пробел»

Определяем индексы глобала



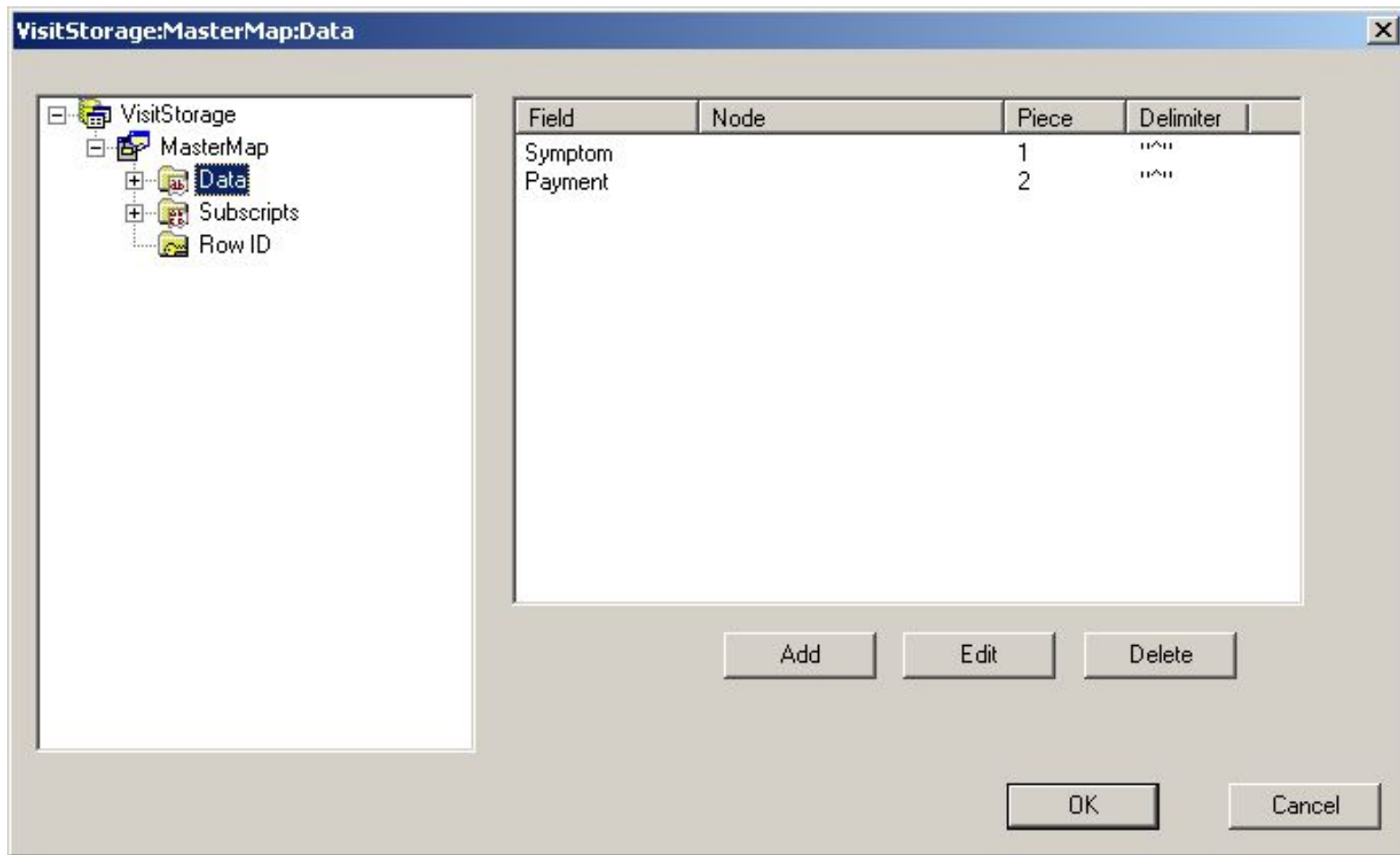
- Первый уровень индекса глобала основан на Training.Patient.SSN
- Второй уровень индекса глобала основан на литерале “Visits”
- Третий уровень индекса глобала основан на VisitDate
- Четвертый уровень индекса глобала основан на VisitTime

Определяем Row ID

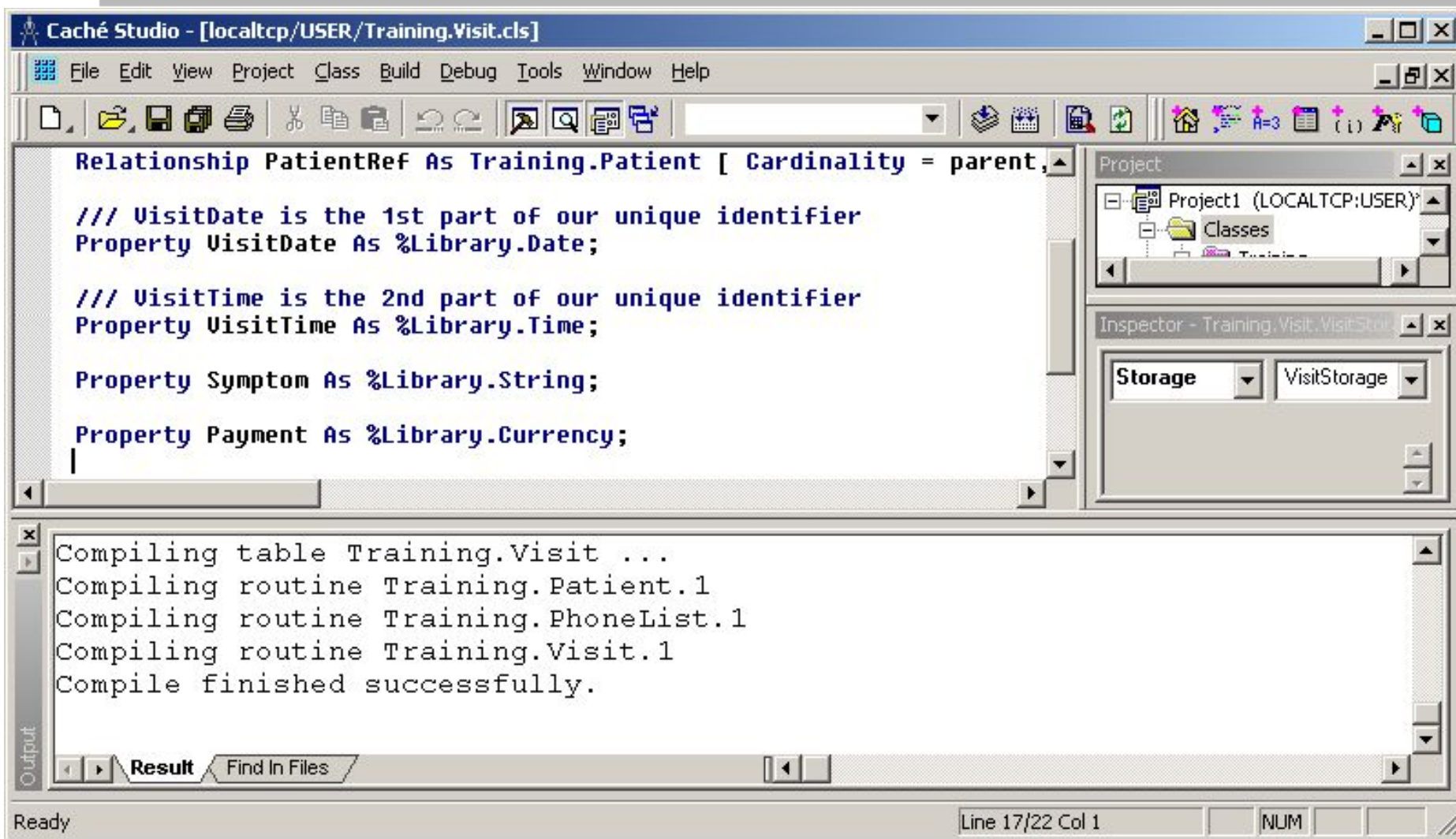


- Row ID 1 основан на свойстве Training.Patient.SSN, которое хранится на первом уровне индекса глобала
- Row ID 2 основан на свойстве VisitDate, которое хранится на третьем уровне индекса глобала
- Row ID основан на свойстве VisitTime, которое хранится на четвертом уровне индекса глобала

Определяем свойства



Сохраняем и компилируем класс





Работа с существующими глобалами через объекты и SQL

Вадим Федоров

InterSystems Corporation