



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

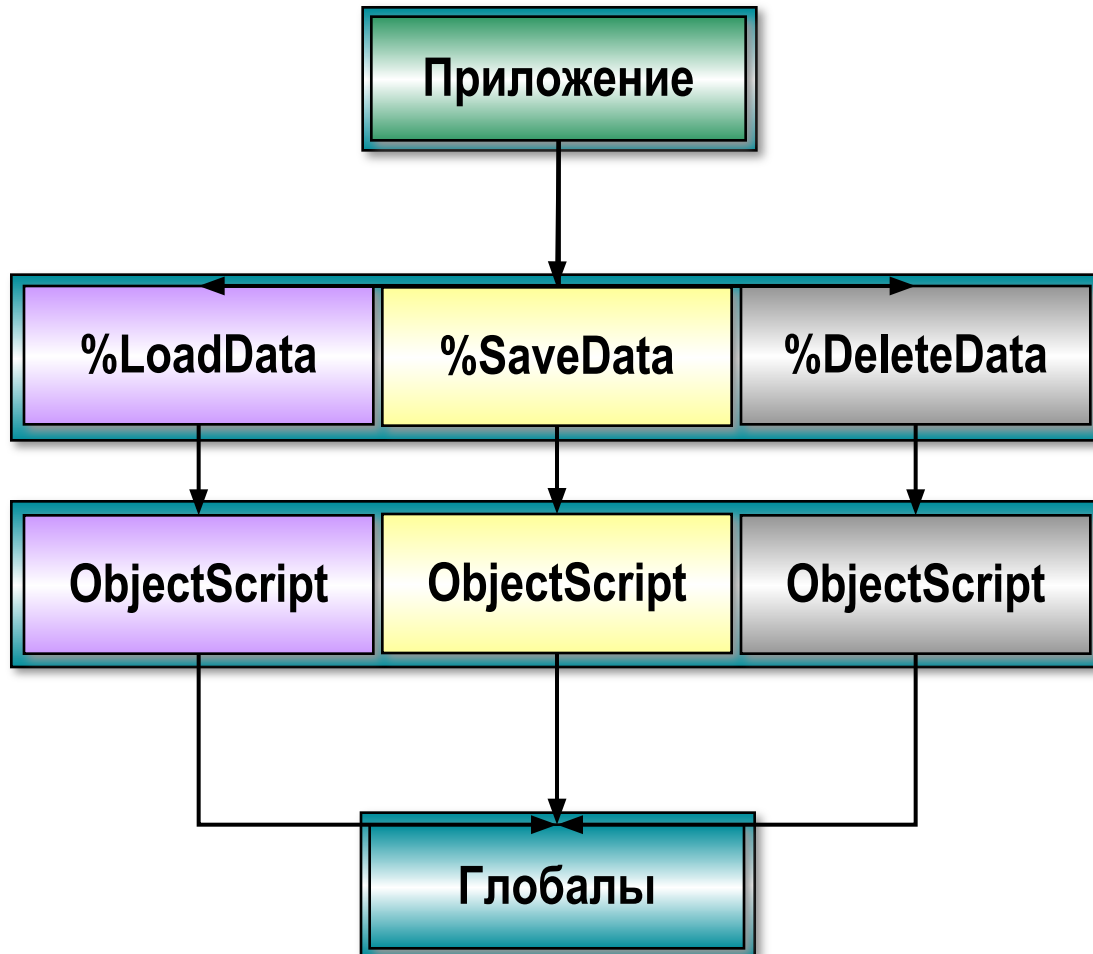
CacheStorage

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

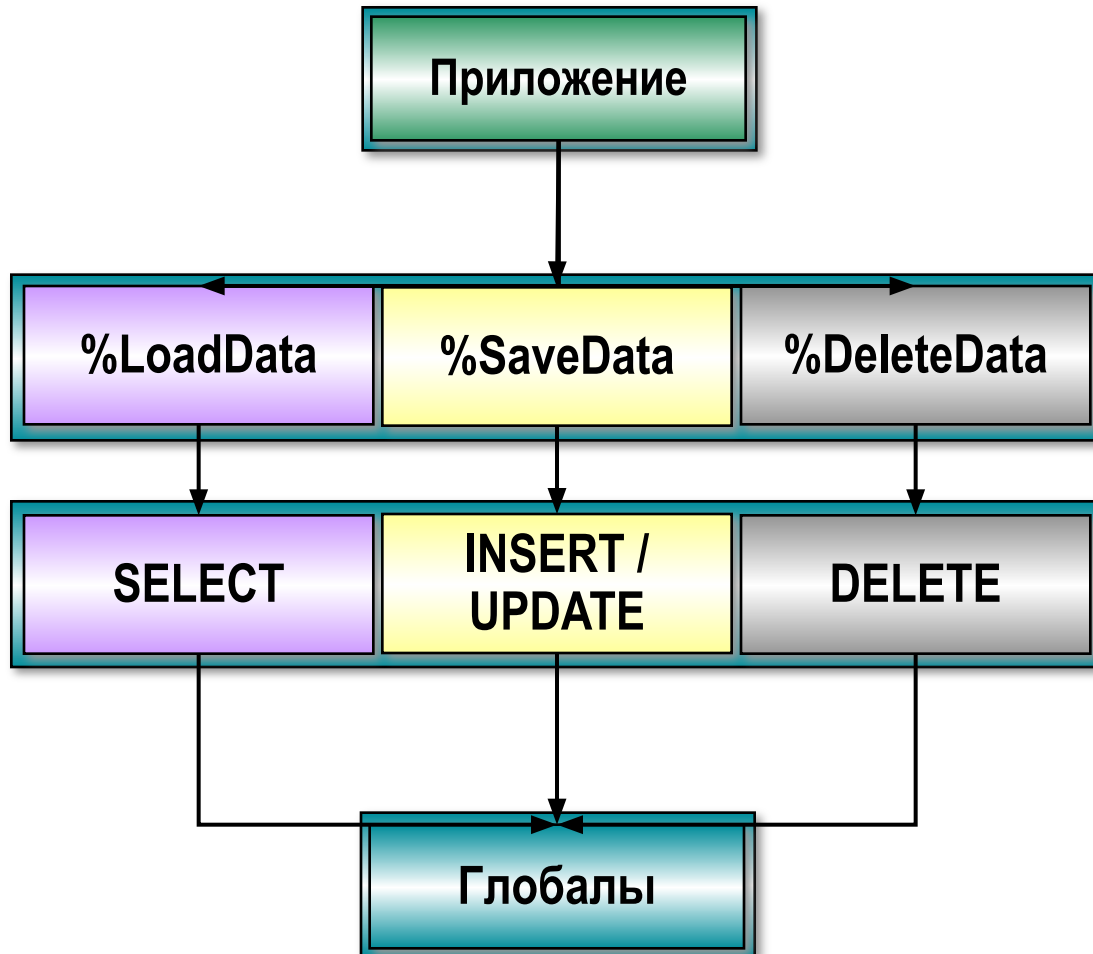
✓
CacheStorage

✓
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

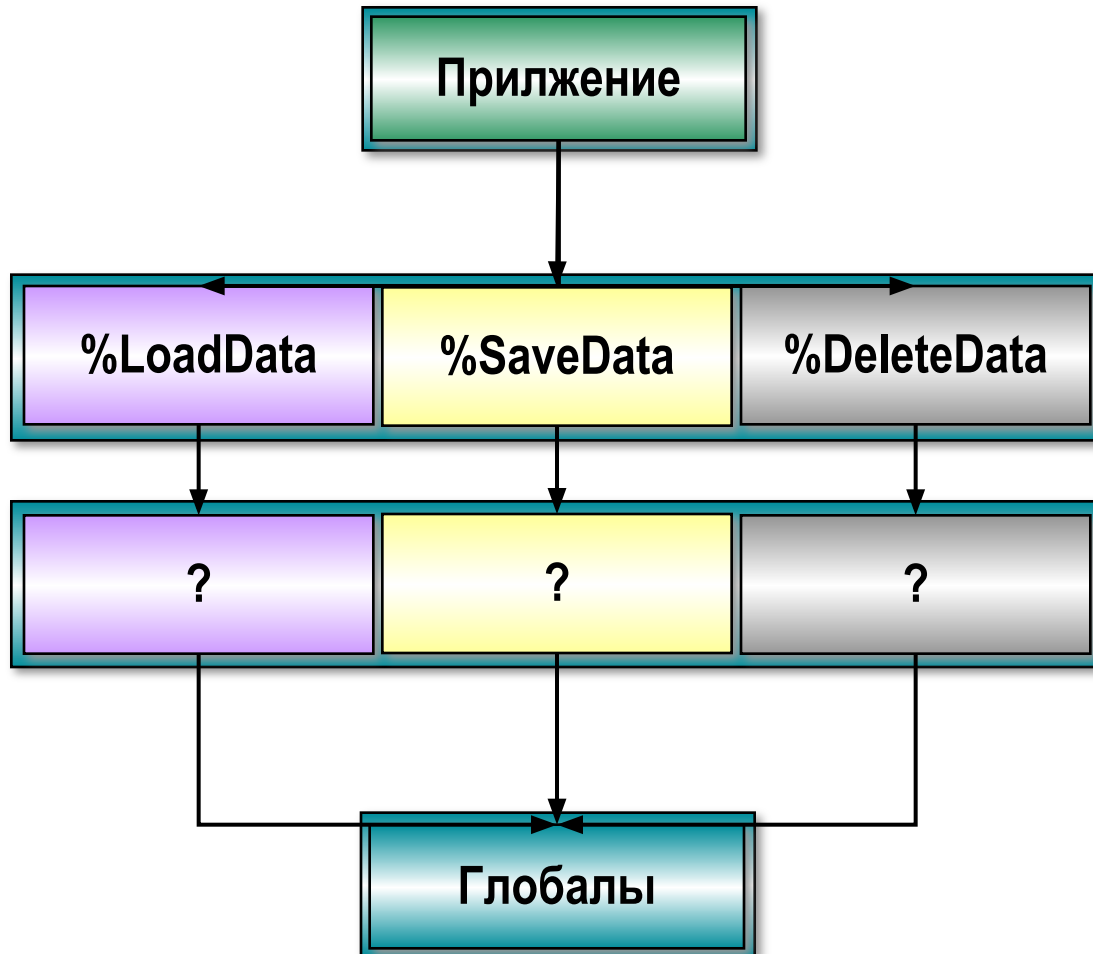
✓
CacheStorage

✓
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

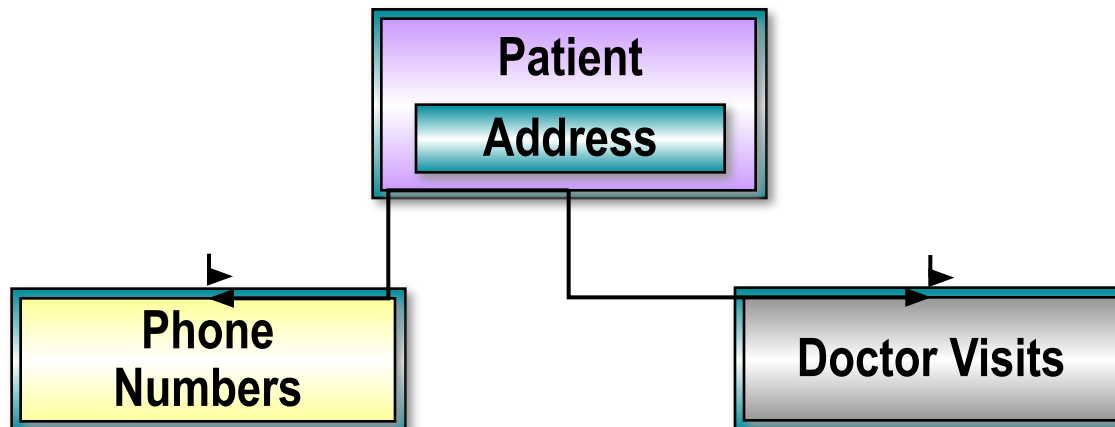
✓
CacheStorage

✓
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

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

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

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

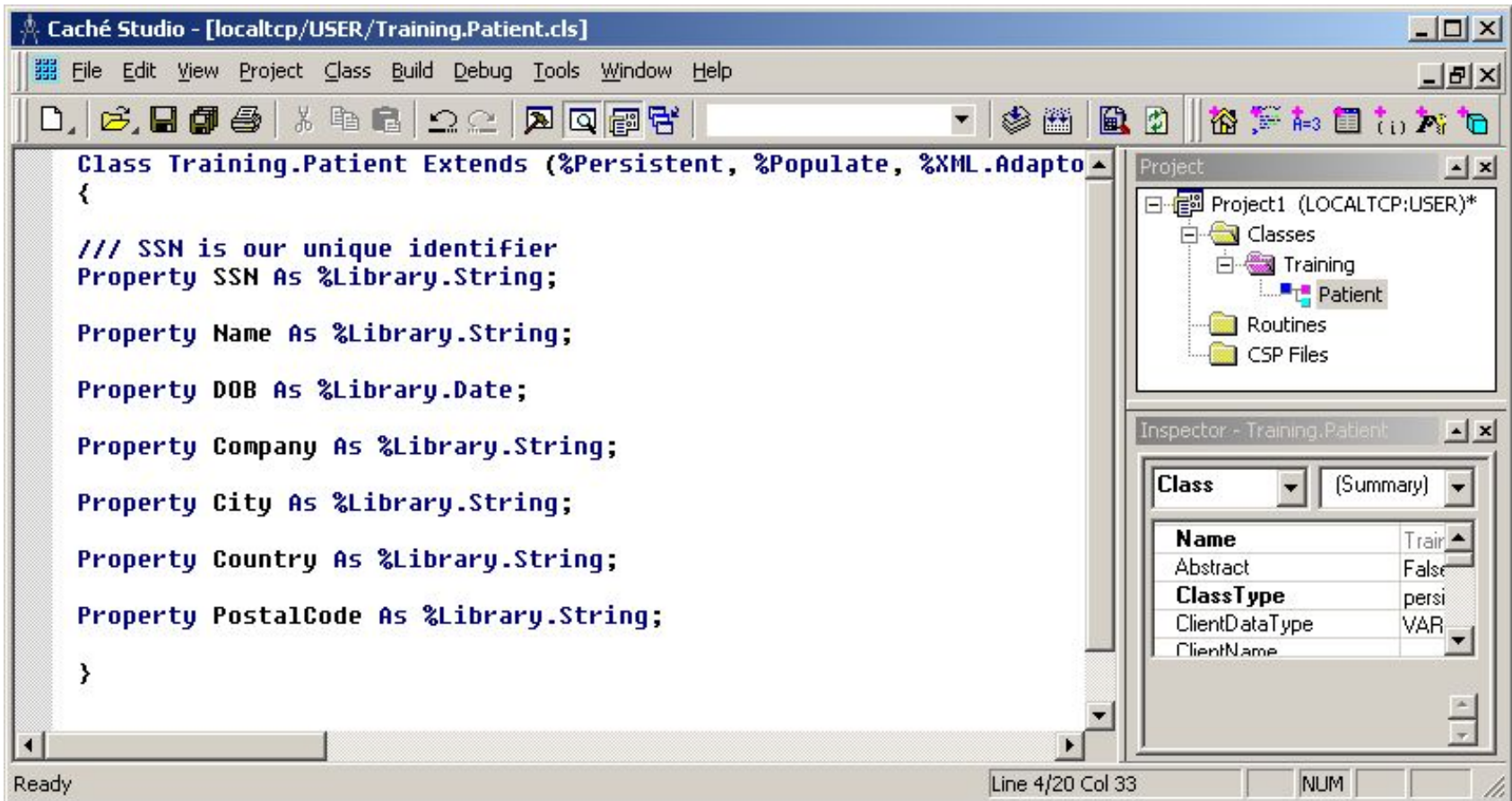
The screenshot shows the Caché Studio IDE with the following components:

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

```
Class Training.Patient Extends (%Persistent, %Populate, %XML.Adapto
{
  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 Explorer:** Shows a tree view with 'Project1 (LOCALTCP:USER)*' containing 'Classes', 'Routines', and 'CSP Files'. Under 'Classes', there is a 'Training' folder containing a 'Patient' class.
- Inspector:** Titled 'Inspector - Training.Patient', it shows a table of class properties:

Class		(Summary)
Name	Train	
Abstract	False	
Class Type	persi	
ClientDataType	VAR	
ClientName		
- Status Bar:** Ready, Line 1/19 Col 1, NUM

Выбираем уникальный идентификатор



```
Class Training.Patient Extends (%Persistent, %Populate, %XML.Adapto
{
    /// 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 Explorer:

- 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

New Storage Wizard

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

< Back Next > Finish Cancel Help

Создаем карту данных

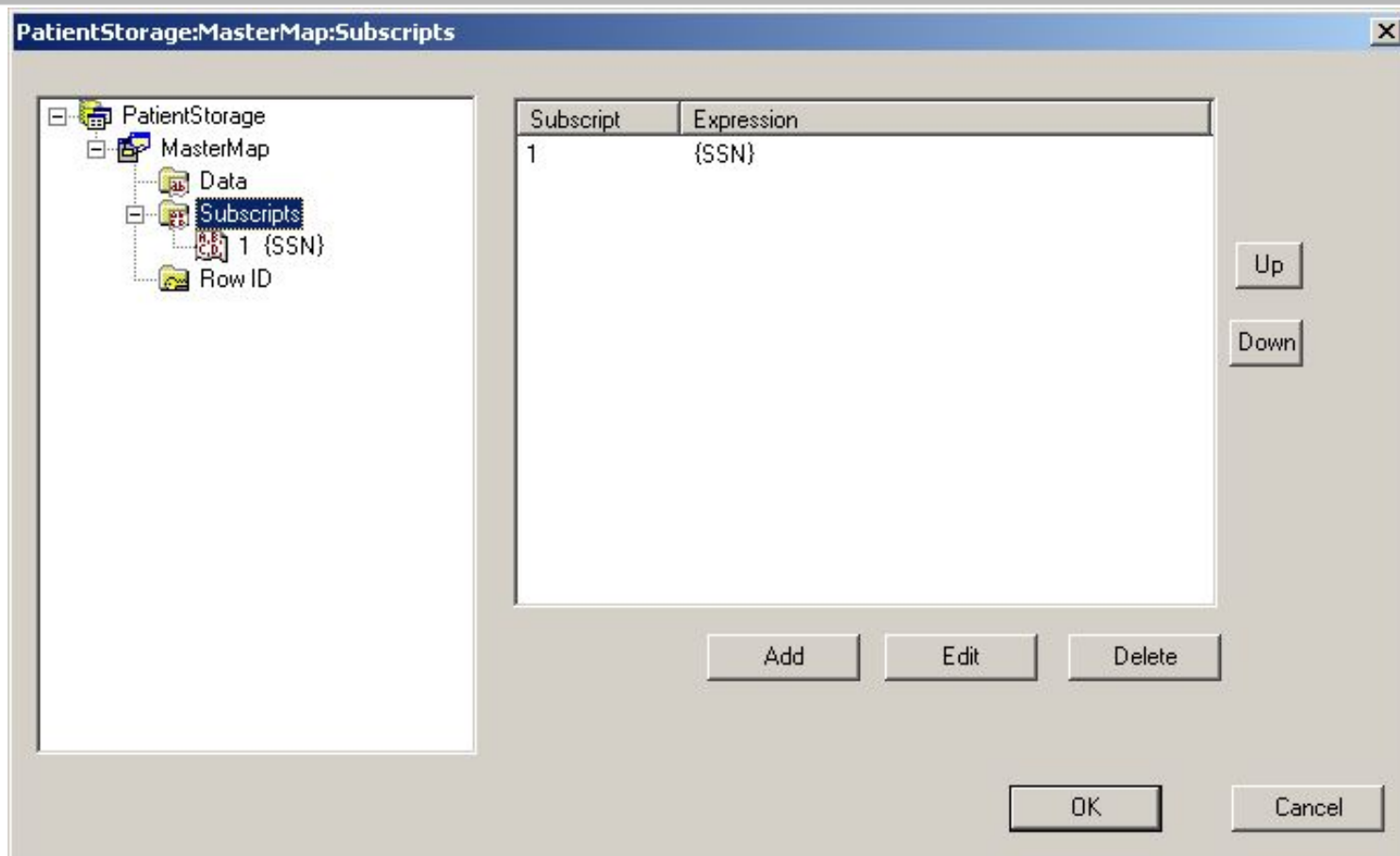
The screenshot shows a dialog box titled "PatientStorage:MasterMap". On the left is a tree view showing a folder "PatientStorage" containing a sub-item "MasterMap". The main area of the dialog contains the following fields:

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

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

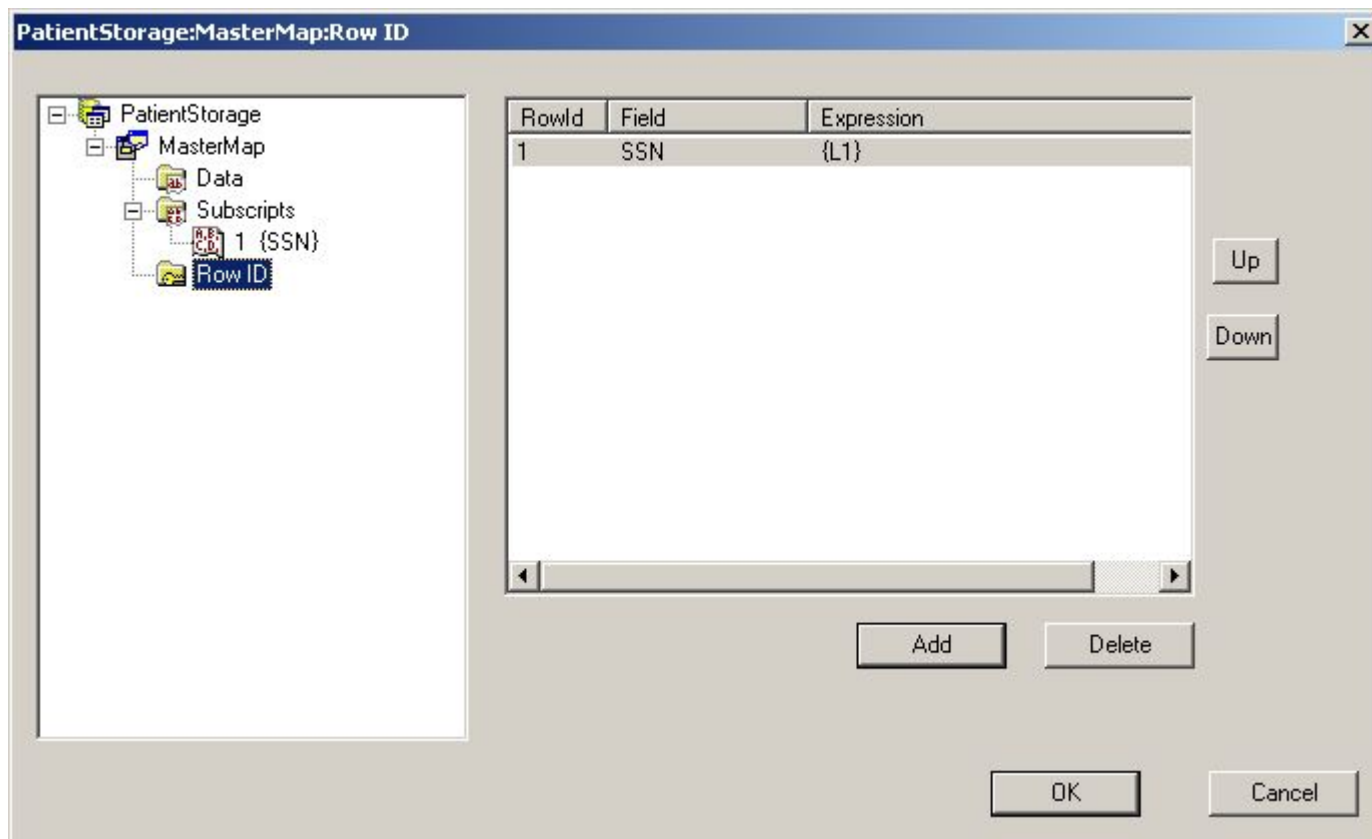
- Имя карты не может содержать символ «пробел»

Определяем индексы глобала



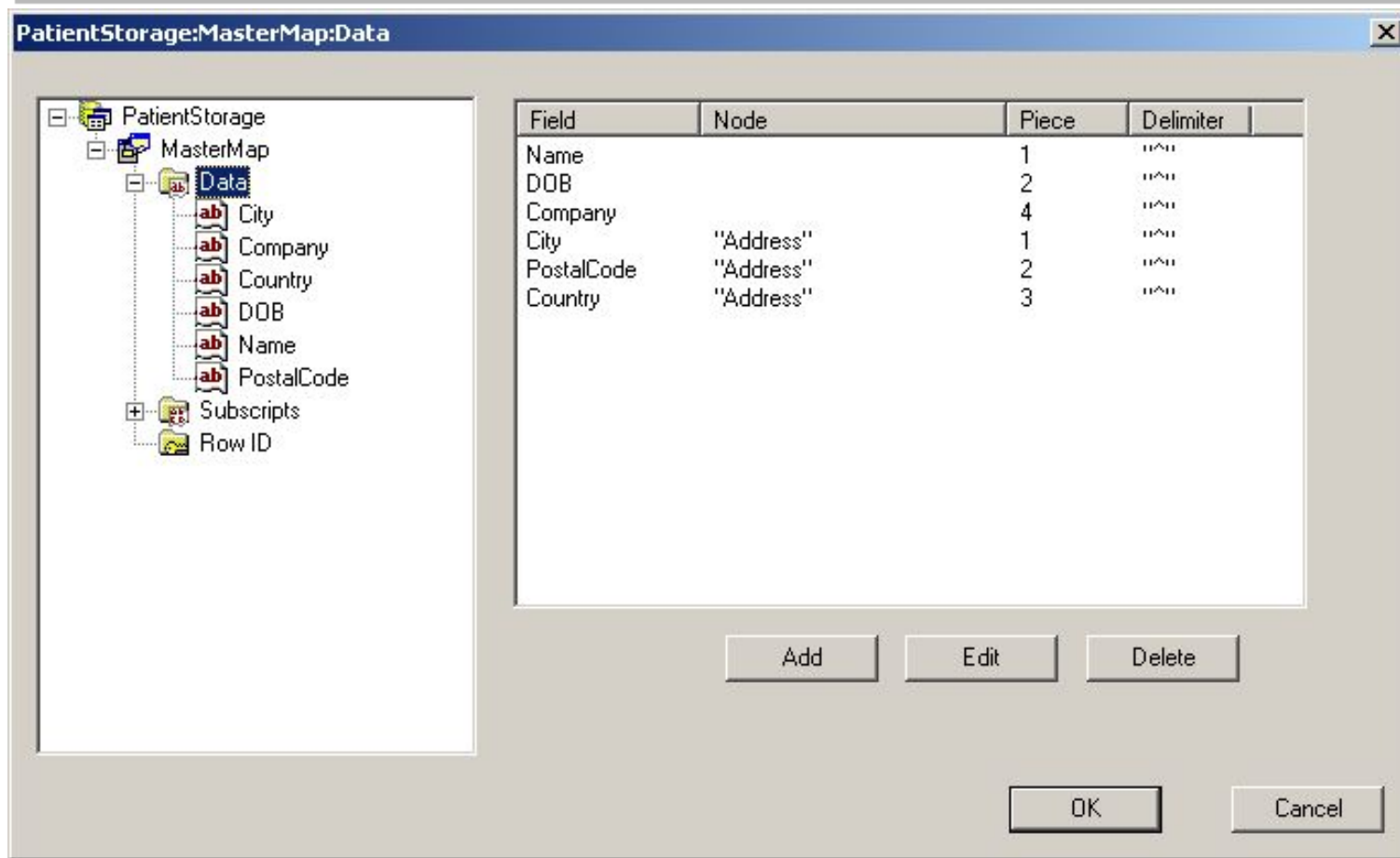
- Первый уровень индекса глобала - SSN

Определяем Row ID



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

Определяем свойства



- Введите разделитель и дополнительную информацию

Срздаем карту индексов

PatientStorage:Map1

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

The screenshot shows a dialog box titled "PatientStorage:NameIndex:Subscripts". On the left is a tree view showing the following structure:

- PatientStorage
 - MasterMap
 - NameIndex
 - Data
 - Subscripts**
 - Row ID

The main area contains a table with the following data:

Subscript	Expression
1	{Name}
2	{SSN}

Control buttons are located on the right and bottom of the dialog:

- Up
- Down
- Add
- Edit
- Delete
- OK
- Cancel

Определяем Row ID индекса

PatientStorage:NameIndex:Row ID

RowId	Field	Expression
1	SSN	{L2}

Up
Down
Add
Delete
OK
Cancel

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

Сохраняем и компилируем класс

The screenshot displays the Caché Studio IDE interface. The main window shows the source code for the class `Training.Patient`, which defines several properties:

```
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;
```

The right-hand side of the IDE features a **Project** pane showing the project structure and an **Inspector** pane for the selected class, displaying the **Storage** property set to `PatientStorage` and the **SQL storage map**.

The **Output** window at the bottom shows the compilation process:

```
Compiled started on 10/30/2002 22:46:50  
Compiling class Training.Patient .....  
Compiling table Training.Patient ...  
Compiling routine Training.Patient.1  
Compile finished successfully.
```

The status bar at the bottom indicates the current cursor position is at **Line 18/23 Col 40**.

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:

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

< 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 following content:

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

The Project pane shows the following structure:

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

The Inspector pane shows the following properties for Training.PhoneList:

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 following components:

- Code Editor:** Contains the following code:

```
/// 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;  
}
```
- Project Explorer:** Shows a project structure with folders for Classes, Routines, and CSP Files. Under Classes, there is a Training folder containing Patient and PhoneList classes.
- Inspector:** Shows the properties of the selected class (Training.PhoneList):

Property	Value
Name	Train
Abstract	False
Class Type	persist
ClientDataType	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

New Storage Wizard

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:

< Back Next > Finish Cancel Help

Создаем карту данных

PatientStorage:MasterMap

Map Name: MasterMap

Map Type: data

Global Name: ^P

Node Structure: \$Piece

Population Type:

Population %:

Condition:

Conditional Fields:

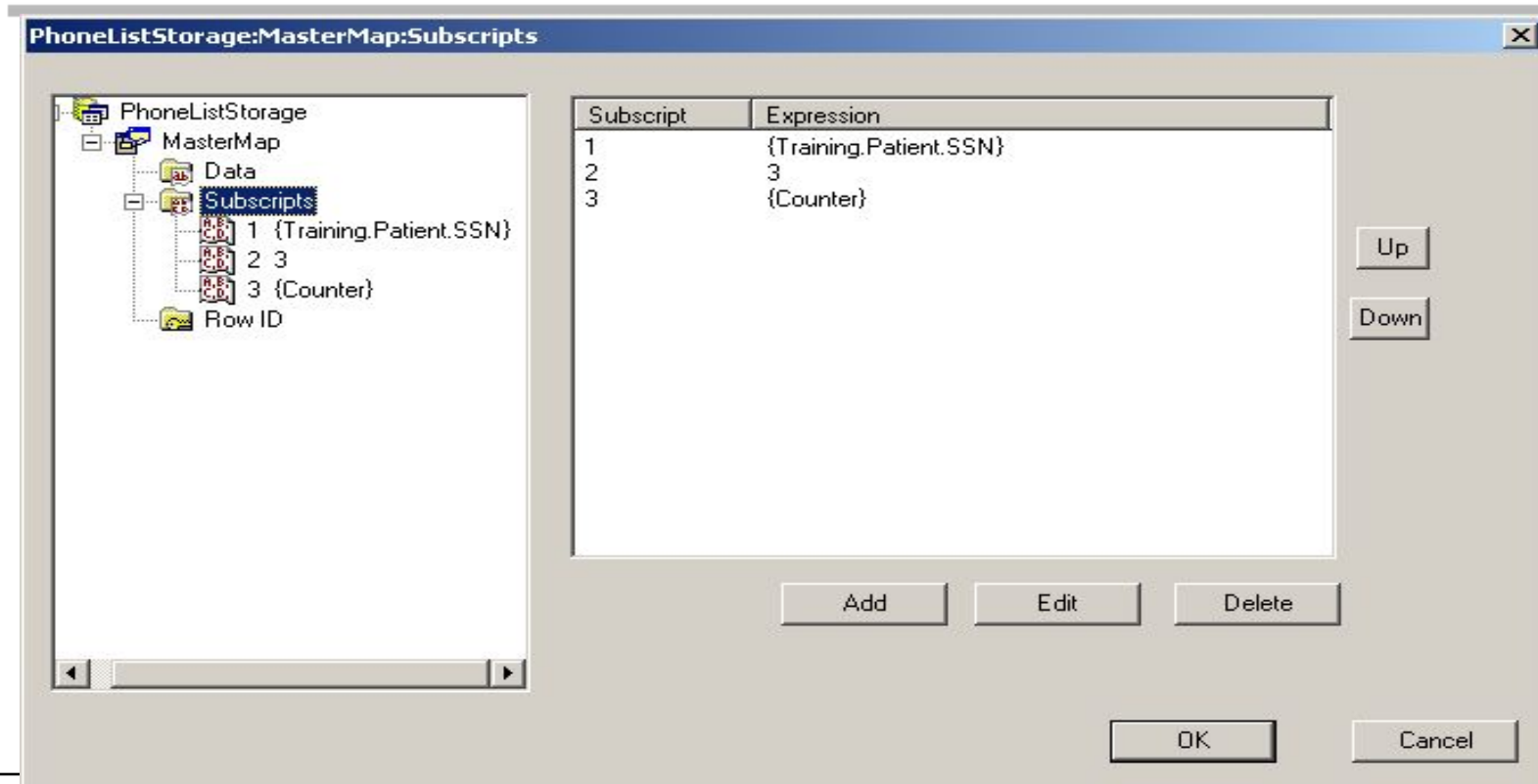
Conditional with hostvars:

Row reference:

OK Cancel

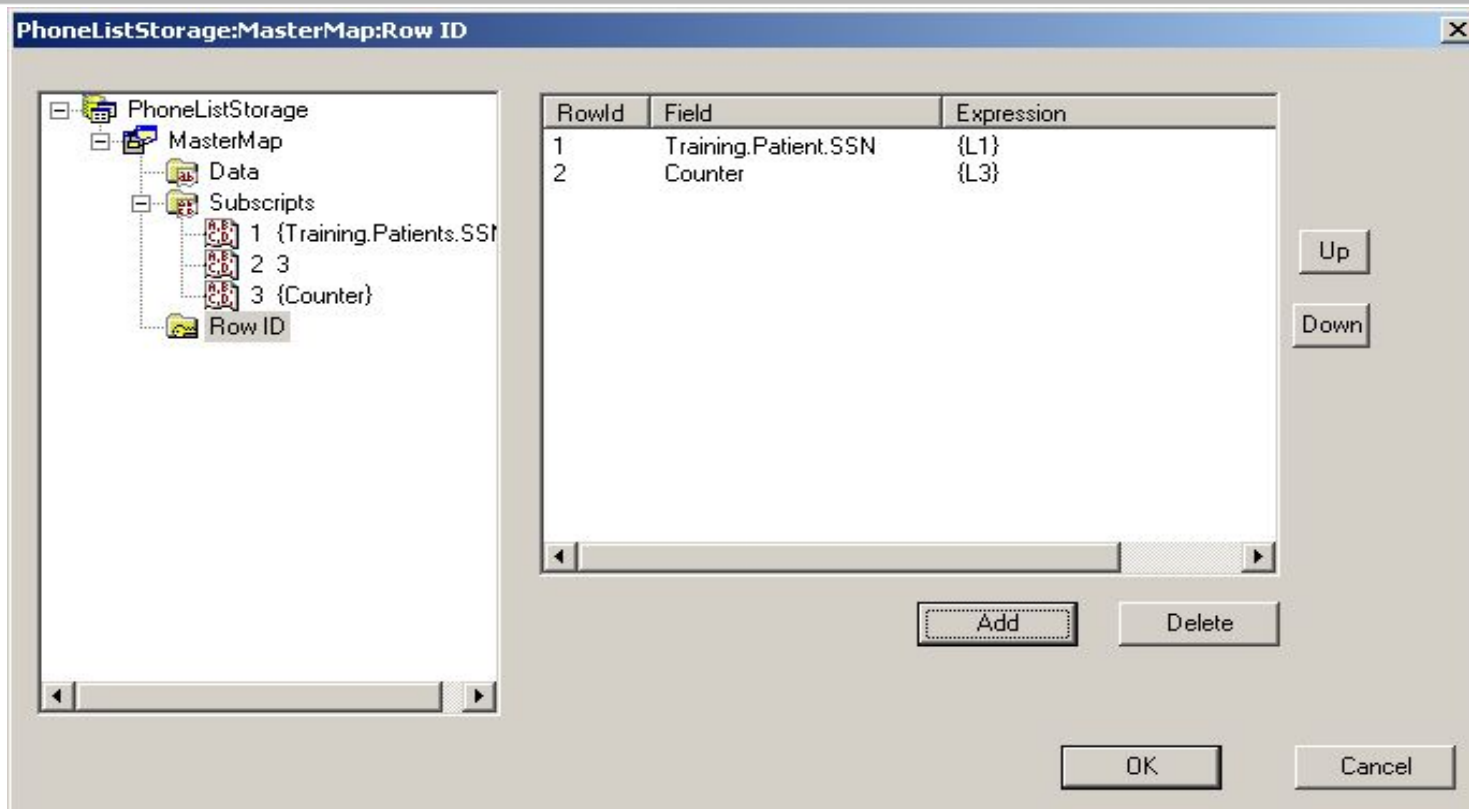
- Имя карты не может содержать символ «пробел»

Определяем индексы глобала



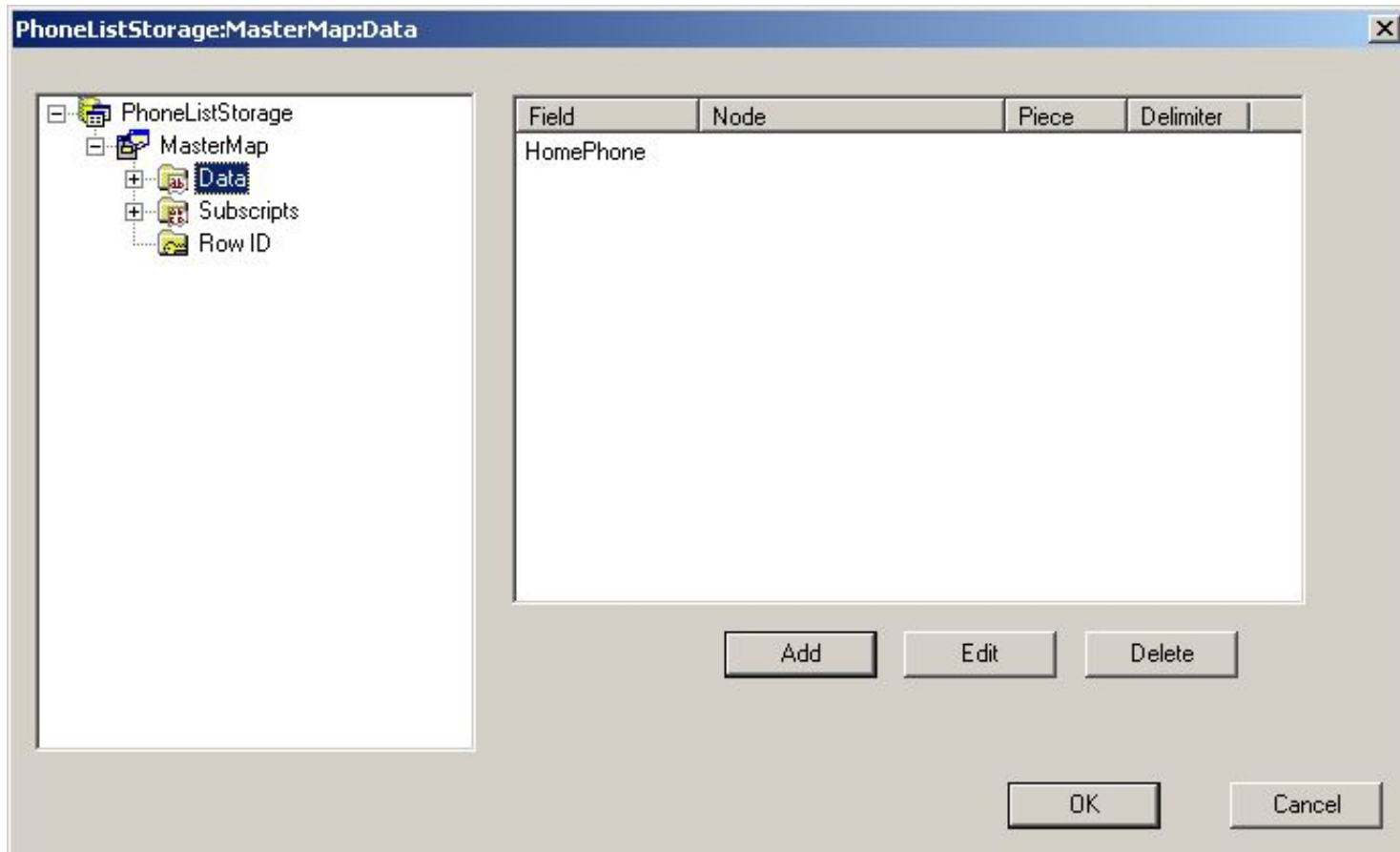
- Первый уровень индекса глобала основан на Training.Patient.SSN
- Второй уровень индекса глобала основан на “^” разделителе, используя третью позицию
- Третий уровень индекса глобала основан на “~” разделителе, используя свойство Counter для позиции

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

The screenshot displays the Caché Studio IDE interface. The main editor window shows the source code for the class `Training.PhoneList`. The code includes comments and declarations for a nested child table, a relationship, and a property.

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

The right-hand side of the IDE features a Project Explorer showing the project structure and an Inspector window for the selected class.

The Output window at the bottom displays the compilation results:

```
Compiling table Training.Patient ...  
Compiling table Training.PhoneList ...  
Compiling routine Training.Patient.1  
Compiling routine Training.PhoneList.1  
Compile finished successfully.
```

The status bar at the bottom indicates the current cursor position: Line 16/16 Col 1.

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:

Enter a class name:

Enter a description of this new class (optional):

< 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

< Back Next > Finish Cancel Help

- 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:

- Source Code:**

```
/// Sub-node child table of Training.Patient  
  
Class Training.Visit Extends (%Persistent, %Populate, %XML.Adaptor) [  
{  
  
Relationship PatientRef As Training.Patient [ Cardinality = parent, ]  
  
Property VisitDate As %Library.Date;  
Property VisitTime As %Library.Time;  
Property Symptom As %Library.String;  
Property Payment As %Library.Currency;  
}
```
- Project Pane:** Shows a tree view for Project1 (LOCALTCP:USER)* with folders for Classes, Routines, and CSP Files. Under Classes, there is a Training folder containing Patient, PhoneList, and Visit.
- Inspector Pane:** Shows the properties for the selected 'Payment' property. The properties are:

Property	Value
XMLNAME	
XMLPROJECTION	
XSDTYPE	decin
Private	False
ReadOnly	False

Ready Line 17/17 Col 1 NUM

Выбираем уникальный идентификатор

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

```
/// Sub-node child table of Training.Patient  
  
Class Training.Visit Extends (%Persistent, %Populate, %XML.Adaptor) [  
{  
  
Relationship PatientRef As Training.Patient [ Cardinality = parent, ]  
  
/// 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;  
  
}  
}
```

The Project Explorer on the right shows the following structure:

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

The Inspector window shows the following properties for the selected class:

Name	Value
Abstract	False
ClassType	persis
ClientDataType	VARC
ClientName	

- 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

New Storage Wizard

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:

< Back Next > Finish Cancel Help

Создаем карту данных

PatientStorage:MasterMap

Map Name: MasterMap

Map Type: data

Global Name: ^P

Node Structure: \$Piece

Population Type:

Population %:

Condition:

Conditional Fields:

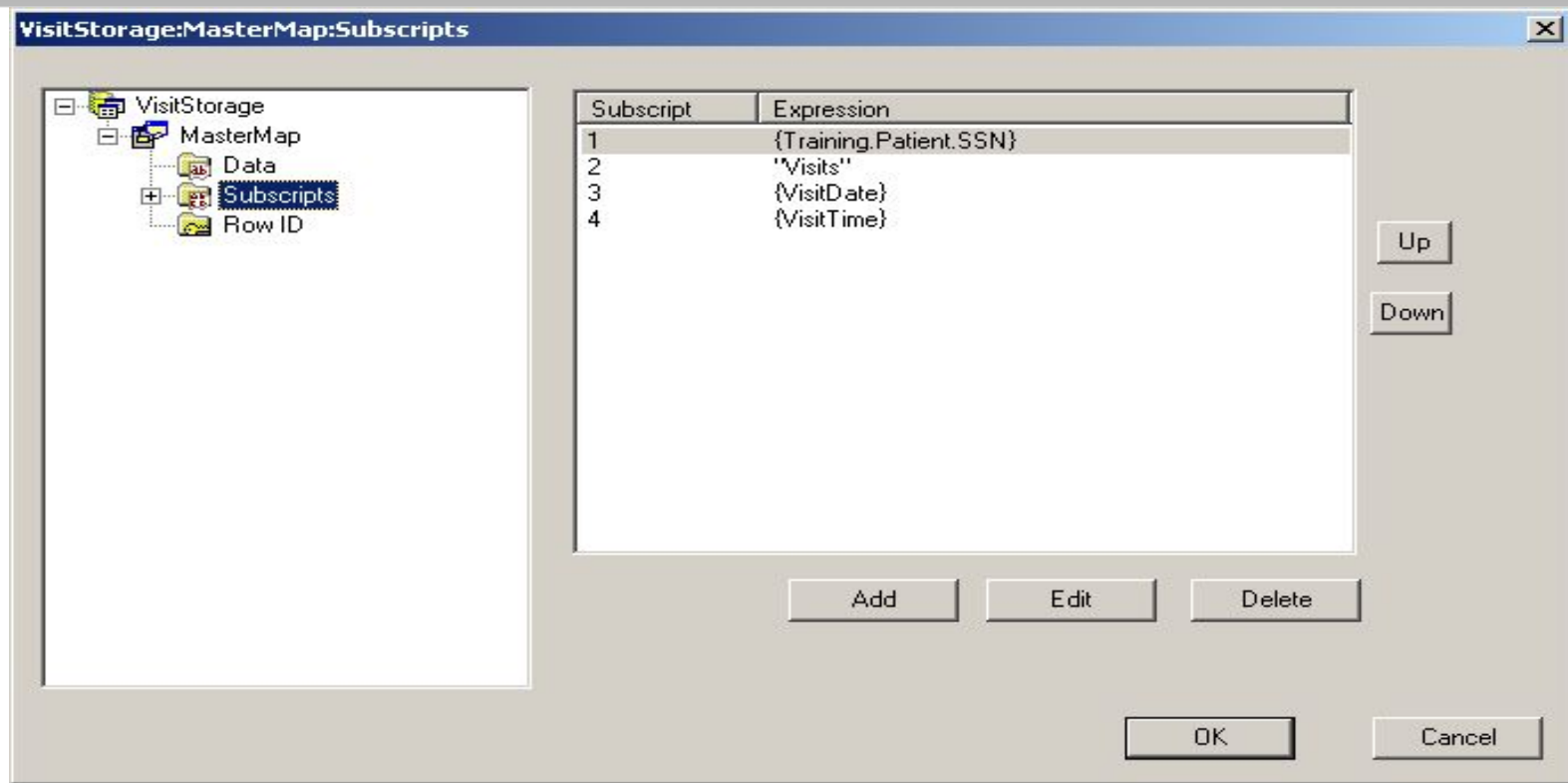
Conditional with hostvars:

Row reference:

OK Cancel

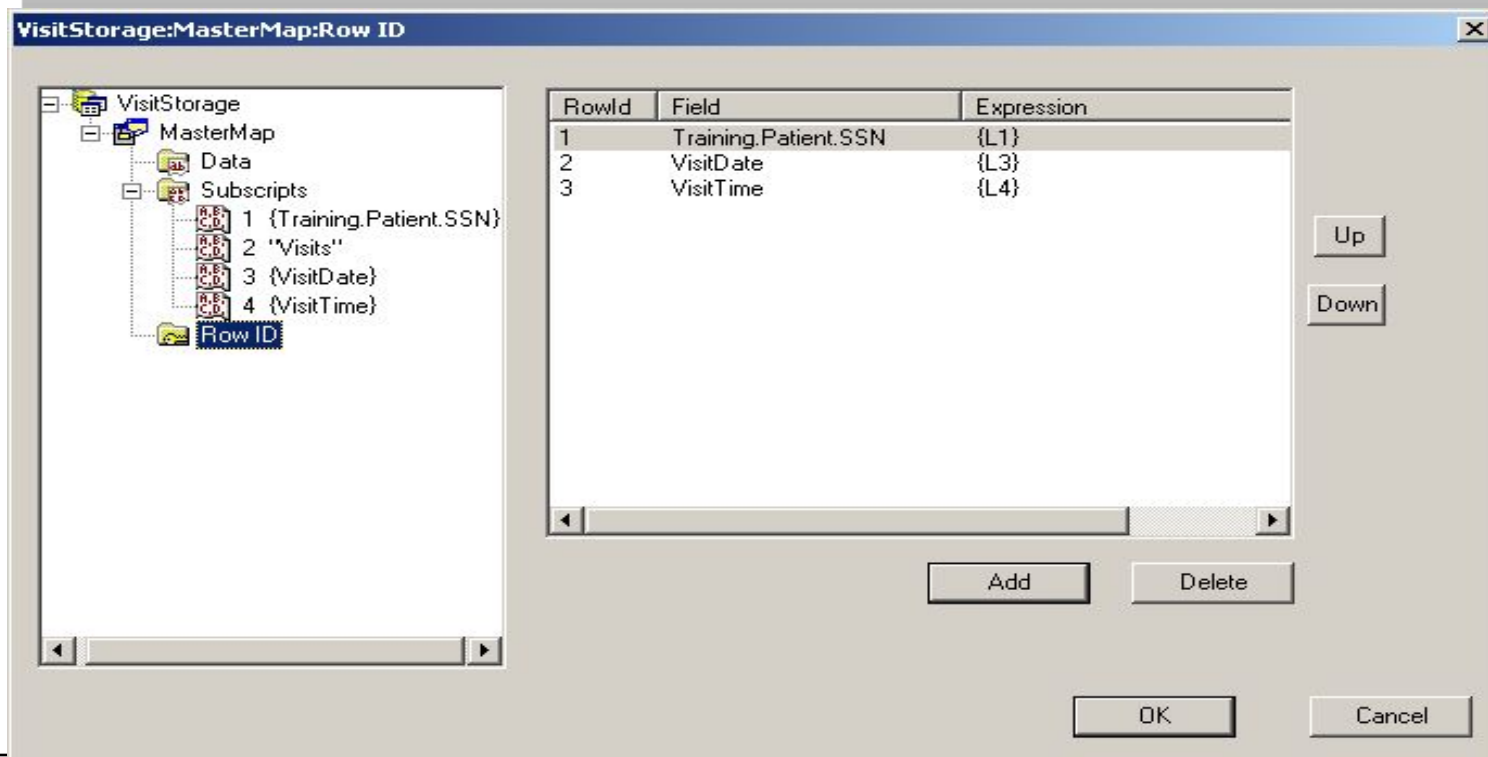
- Имя карты не может содержать символ «пробел»

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

VisitStorage:MasterMap:Data

VisitStorage
MasterMap
Data
Subscripts
Row ID

Field	Node	Piece	Delimiter
Symptom		1	""
Payment		2	""

Add Edit Delete

OK Cancel

Сохраняем и компилируем класс

The screenshot displays the Caché Studio IDE interface. The main editor window shows the following code:

```
Relationship PatientRef As Training.Patient [ Cardinality = parent,  
  
/// 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;
```

The Project Explorer on the right shows a project named "Project1 (LOCALTCP:USER)" containing a folder "Classes". The Inspector panel shows the "Storage" property set to "VisitStorage".

The Output window at the bottom displays the following compilation log:

```
Compiling table Training.Visit ...  
Compiling routine Training.Patient.1  
Compiling routine Training.PhoneList.1  
Compiling routine Training.Visit.1  
Compile finished successfully.
```

The status bar at the bottom indicates "Ready" and "Line 17/22 Col 1".



Работа с существующими глобалами через объекты и SQL

Вадим Федоров

InterSystems Corporation