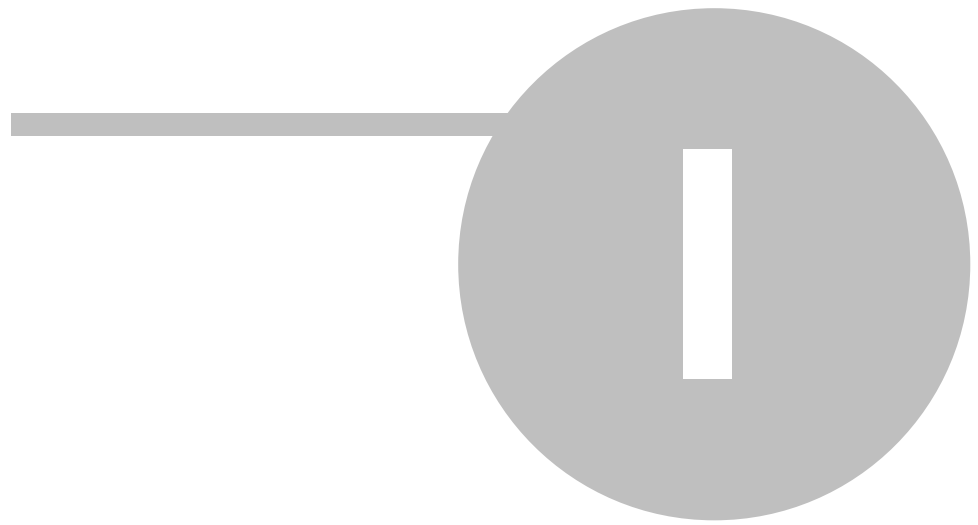


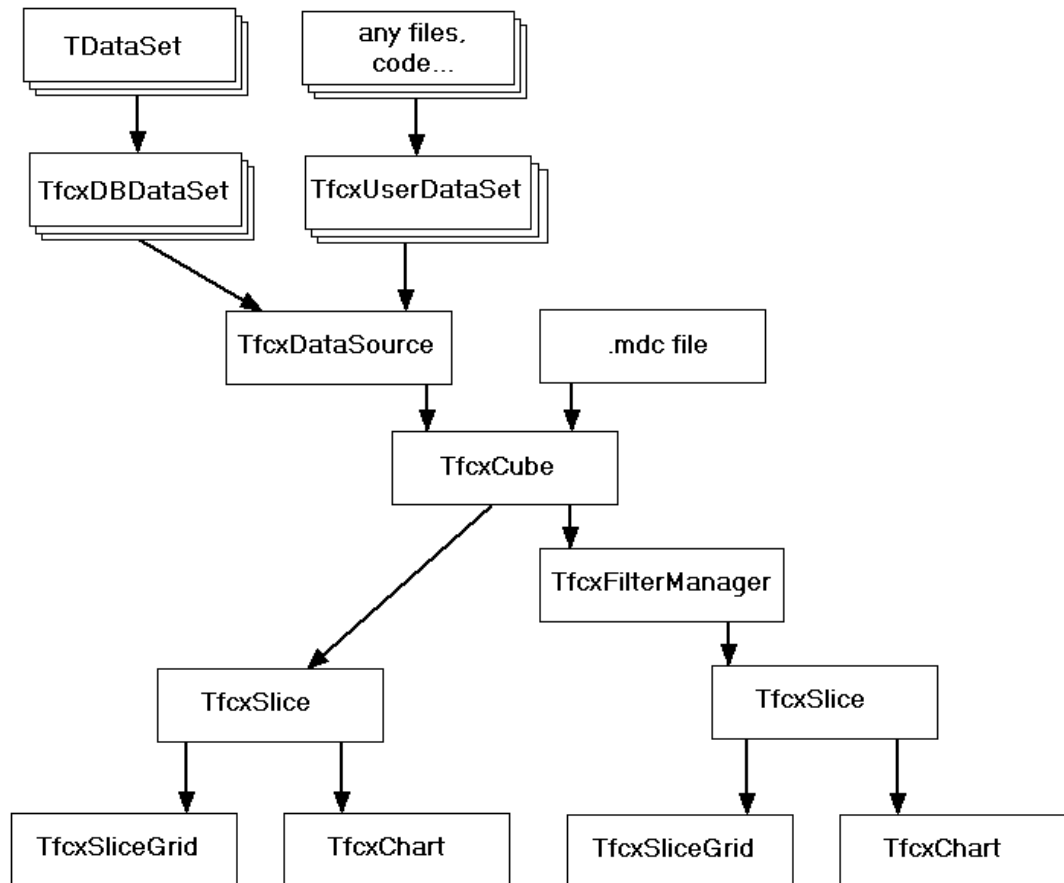
FastCube 2.0

		I
I	FastCube 2	2
II	FastCube	5
1	5
	5
	6
	7
2	8
	9
	TfcxDataSource	10
	TfcxDataSource	11
3	12
	13
	15
4	16
5	17



FastCube 2

Библиотека компонент FastCube 2 - это набор невидимых и визуальных компонент, назначение которых - обработка, хранение и визуализация многомерных данных. Архитектура компонент FastCube 2 представлена на следующей диаграмме.



FastCube 2 включает в себя следующие компоненты:

Невидимые компоненты

- **TfcxDBDataSet** - компонент для связи с источником из базы данных;
- **TfcxUserDataSet** - компонент для связи с пользовательским источником (на основе событий);
- **TfcxDataSource** - компонент связывающий все источники данных для куба и описывающий свойства полей и атрибутов;
- **TfcxCube** - (куб) основное хранилище загруженных данных;
- **TfcxSlice** - (срез) отвечает за хранение структуры представления данных и их подготовку в соответствии с ней;
- **TfcxFilterManager** - управляет фильтрацией данных куба для среза;

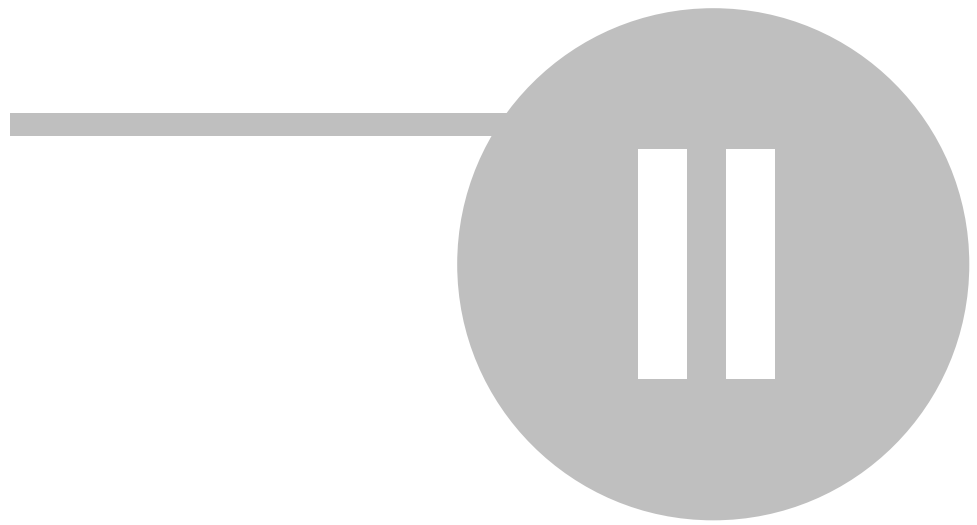
Базовые визуальные компоненты:

- **TfcxCubeGrid** - (таблица данных) отображает исходные данные из куба;

- **TfcxSliceGrid** - (кросс-таблица) отображает данные в соответствии со структурой из среза и предоставляет пользователям возможность манипулирования данными и структурой;
- **TfcxCubeGridToolbar** - (панель инструментов таблицы данных) содержит набор кнопок, позволяющих выполнять операции с таблицей данных;
- **TfcxSliceGridToolbar** - (панель инструментов кросс-таблицы) содержит набор кнопок, позволяющих выполнять операции с кросс-таблицей, срезом и кубом;

Компоненты для отображения диаграмм:

- **TfcxChart** - (диаграмма) отвечает за представление данных в виде диаграммы.
- **TfcxChartToolbar** - (панель инструментов диаграммы) содержит набор кнопок, позволяющих выполнять операции с диаграммой.



FastCube

2.1

Загруженные в куб данные можно сохранить для последующего использования. Сохраненные данные можно будет загружать в куб без обращения к базе данных источника. Кроме самих данных можно сохранять настройку среза, групп, фильтров и диаграмм. Сохранить куб и/или настройки можно в файле или в BLOB-поле БД. Для это необходимо использовать методы компонентов TfcxCube, TfcxSlice, TfcxFilterManager и TfcxChart.

2.1.1

Данные куба (TfcxCube):

function LoadFromFile(ACubeFileName: String): Boolean;

Загружает данные куба из файла с заданным именем. Если файл загружен успешно, возвращает True.

Куб очищается перед загрузкой данных.

function LoadFromStream(ACubeStream: TStream): Boolean;

Загружает данные куба из потока. Если данные загружены успешно, возвращает True.

Куб очищается перед загрузкой данных.

function AppendFromFile(ACubeFileName: String): Boolean;

Загружает данные куба из файла с заданным именем, добавляя их к загруженным. Если файл загружен успешно, возвращает True.

Происходит объединение данных.

function AppendFromStream(ACubeStream: TStream): Boolean;

Загружает данные куба из потока, добавляя их к загруженным. Если данные загружены успешно, возвращает True.

Происходит объединение данных.

procedure SaveToFile(ACubeFileName: String; AFilter: TObject = nil);

Записывает данные куба в файл с заданным именем. Если в параметре AFilter указан объект TfcxFilterManager, то записываются только допущенные данным фильтром записи.

procedure SaveToStream(ACubeStream: TStream; ACompressionLevel: TCompressionLevel = clMax; AFilter: TObject = nil);

Записывает данные куба в поток. Если в параметре AFilter указан объект TfcxFilterManager, то записываются только допущенные данным фильтром записи. Параметр ACompressionLevel указывает степень сжатия.

Вместе с данными в файл куба записываются настройки групп и схемы всех подключенных срезов. Состояние фильтров и настройки подключенных диаграмм вместе со схемами срезов не сохраняются.

Файл с данными куба имеет по умолчанию расширение mdc.

Примеры:

```
fcxCube1.LoadFromFile('c:\cube1.mdc');
fcxCube1.AppendFromFile('c:\cube1.mdc');
fcxCube1.SaveToFile('c:\cube2.mdc');
fcxCube1.SaveToFile('c:\cube2Filter.mdc', fcxFilterManager1);
```

2.1.2

Настройка среза (TfcxSlice):

function LoadFromFile(AFileName: String): Boolean;

Загружает схему среза из файла с заданным именем. Если файл загружен успешно, возвращает True.

Схема среза сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о группах, настройка групп в кубе сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о фильтрах, настройка фильтров в менеджере фильтров сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о настройках диаграмм, настройка подключенных диаграмм сбрасывается перед загрузкой.

function LoadFromStream(ASliceStream: TStream): Boolean;

Загружает схему среза из потока. Если данные загружены успешно, возвращает True.

Схема среза сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о группах, настройка групп в кубе сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о фильтрах, настройка фильтров в менеджере фильтров сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о настройках диаграмм, настройка подключенных диаграмм сбрасывается перед загрузкой.

procedure SaveToFile(AFileName: String; AStoreItems: TfcxItemsForStoreWithSlice = []);

Записывает схему среза в файл с заданным именем. Параметр AStoreItems описывает, какую информацию надо сохранять дополнительно (фильтры, группы, диаграммы).

procedure SaveToStream(ASliceStream: TStream; AStoreItems: TfcxItemsForStoreWithSlice = []);

Записывает схему среза в поток. Параметр AStoreItems описывает, какую информацию надо сохранять дополнительно (фильтры, группы, диаграммы).

Вместе со схемой в файл среза могут быть записаны настройки групп, фильтров и диаграмм. Файл схемы среза имеет xml структуру, расширение по умолчанию - mds.

Примеры:

```
fcxSlicel1.LoadFromFile('c:\schema1.mds');  
fcxSlicel1.SaveToFile('c:\schema2.mds');  
fcxSlicel1.SaveToFile('c:\schema3.mds', [fcxiss_Groups, fcxiss_Filters,  
fcxiss_Charts]);
```

2.1.3

Настройка фильтров (TfcxFilterManager):

function LoadFromFile(AFileName: String): Boolean;

Загружает настройку фильтров из файла с заданным именем. Если файл загружен успешно, возвращает True.

Настройка фильтров сбрасывается перед загрузкой.

function LoadFromStream(AStream: TStream): Boolean;

Загружает настройку фильтров из потока. Если данные загружены успешно, возвращает True.

Настройка фильтров сбрасывается перед загрузкой.

procedure SaveToFile(AFileName: String);

Записывает настройку фильтров в файл с заданным именем.

procedure SaveToStream(AStream: TStream);

Записывает настройку фильтров в поток.

Настройка групп (TfcxCube):

function LoadGroupsFromFile(AGroupsFileName: String): Boolean;

Загружает настройку групп из файла с заданным именем. Если файл загружен успешно, возвращает True.

Настройка групп сбрасывается перед загрузкой.

function LoadGroupsFromStream(AStream: TStream): Boolean;

Загружает настройку групп из потока. Если данные загружены успешно,

возвращает True.

Настройка групп сбрасывается перед загрузкой.

procedure SaveGroupsToFile(AGroupsFileName: String);

Записывает настройку групп в файл с заданным именем.

procedure SaveGroupsToStream(AStream: TStream);

Записывает настройку групп в поток.

Настройка диаграмм (TfcxChart):

function LoadFromFile(AFileName: String): Boolean;

Загружает настройку диаграмм из файла с заданным именем. Если файл загружен успешно, возвращает True.

Настройка диаграмм сбрасывается перед загрузкой.

function LoadFromStream(AStream: TStream): Boolean;

Загружает настройку диаграмм из потока. Если данные загружены успешно, возвращает True.

Настройка диаграмм сбрасывается перед загрузкой.

procedure SaveToFile(AFileName: String);

Записывает настройку диаграмм в файл с заданным именем.

procedure SaveToStream(AStream: TStream);

Записывает настройку диаграмм в поток.

Файл с настройкой диаграмм имеет по умолчанию расширение mdt.

Примеры:

```
fcxFilterManager1.LoadFromFile('c:\Filter1.fcf');  
fcxFilterManager1.SaveToFile('c:\Filter2.fcf');
```

```
fcxCube1.LoadGroupsFromFile('c:\Group1.fcg');  
fcxCube1.SaveGroupsToFile('c:\Group2.fcg');
```

```
fcxChart1.LoadFromFile('c:\Chart1.mdt');  
fcxChart1.SaveToFile('c:\Chart2.mdt');
```

2.2

Загрузка данных из БД и пользовательских источников.

2.2.1

Набор компонентов FastCube предназначен для построения сводных таблиц на основе "плоских" данных.

Простейшим источником для куба является таблица базы данных.

Для загрузки данных в куб необходимо подключение к базе через компонент - наследник TDataSet. Его выбор зависит от используемых в приложении компонентов доступа к БД .

TfxDBDataSet служит для связи компонента доступа с описателем источника данных для куба: TfcdDataSource.

TfcdDataSource содержит в себе полное описание структуры данных, загружаемых в куб. В нём описываются все источники, поля этих источников, взаимосвязи между источниками, правила конвертирования данных и т.п. Один из источников является основным, он указывается в свойстве TfcdDataSource.DataSet. В случае, когда загружаем данные только из одной таблицы, достаточно указать только основной источник.

Далее надо связать компоненты TfcdDataSource - TfcdCube - TfcdSlice - TfcdSliceGrid. TfcdCube и TfcdSlice можно связывать через менеджер фильтров TfcdFilterManager, если он упущен, то в TfcdSlice автоматически создастся внутренний объект TfcdFilterManager. Отдельно выделять TfcdFilterManager нужно только в случае, если один менеджер фильтров будет использоваться в нескольких срезах.

В TfcdDataSource можно описать поля, загружаемые из источника. Если описание полей отсутствует, то будут автоматически загружаться все поля.

Описание полей основного источника хранит свойство TfcdDataSource.Fields. Очистить описание можно методом TfcdDataSource.DeleteFields. Заполнить описание полей из источника можно методом TfcdDataSource.AddFields, но заполнение происходит только если в источнике созданы объекты полей (определены в дизайнерах или созданы автоматически при открытии DataSet). Если Вы не планируете менять описание полей, вызывать AddFields не нужно, т.к. его вызов произойдет автоматически при открытии источника данных.

В качестве источника для куба могут служить: компонент TfcdDataSource, файл с сохранением куба, поток. Выбор источника происходит на основании свойства TfcdCube.CubeSource: TfcdCubeSource.

```
TfcdCubeSource =  
(  
    fccs_None,           // None  
    fccs_DataSource,    // load from fcdDataSource  
    fccs_CubeFile,      // load from file  
    fccs_CubeStream     // load from Stream  
);
```

В нашем случае надо присвоить fccs_DataSource.

Загрузка данных куба вызывается методом TfcdCube.Open. При этом куб автоматически открывает необходимые источники и загружает из них данные из описанных полей.

После загрузки данных кросс-таблица готова к работе с пользователем.

Пример:

```
//
fcxDBDataSet1 := TfcxDBDataSet.Create(Self);
fcxDataSource1 := TfcxDataSource.Create(Self);
fcxCube1 := TfcxCube.Create(Self);
fcxSlice1 := TfcxSlice.Create(Self);
fcxSliceGrid1 := TfcxSliceGrid.Create(Self);
fcxSliceGrid1.Parent := Self;
fcxSliceGrid1.Align := alClient;

//
fcxDBDataSet1.DataSet := DataSet1;
fcxDataSource1.DataSet := fcxDBDataSet1;
fcxCube1.DataSource := fcxDataSource1;
fcxSlice1.Cube := fcxCube1;
fcxSliceGrid1.Slice := fcxSlice1;

//
fcxDataSource1.DeleteFields;

//
fcxCube1.CubeSource := fccs_DataSource;

//
fcxCube1.Open;
```

2.2.2

TfcxDataSource

Настройка полей источника необходима в случае, если нужно загружать только часть полей, конвертировать данные, управлять разбивкой полей типа дата-время на составляющие, загружать данные из нескольких связанных источников.

Описание полей основного источника хранит свойство `TfcxDataSource.Fields`. Поле описывается объектом `TfcxSourceField`.

Свойства поля источника `SourceFieldProperties` зависят от его типа `SourceFieldType`: `TfcxAttributeType`.

```
TfcxAttributeType =
(
    fcxsft_Reference, // Поле из источника
    fcxsft_Custom,   // Пользовательское поле
    fcxsft_Date,     // Поле типа Дата (разбивается на составляющие)
    fcxsft_Time      // Поле типа Время (разбивается на составляющие)
);
```

Свойство `DataField` описывает тип данных источника, имя и название поля в источнике, необходимость конвертирования с указанием целевого типа данных, имя и название результирующего поля в кубе. Класс свойства `DataField` зависит от `SourceFieldType`.

Примеры:

```
//
fcxDataSource1.AddFields;

//
fcxDataSource1.Fields[2].DataField.CubeFieldDisplayLabel :=
'          ';

//
                                'Population'.
TfcxReferenceDataField(fcxDataSource1.Fields.FieldName
['Population'].DataField).Convert := True;
TfcxReferenceDataField(fcxDataSource1.Fields.FieldName
['Population'].DataField).CubeFieldType := fcdt_String;
```

2.2.3

TfcxDataSource

SplitProperty: TfcxSplitProperty описывает разбивку поля на составляющие "атрибуты". Атрибутами являются части полей даты и времени (год, день, час и т.д.) и поля зависимых источников. Атрибут в свою очередь может иметь собственные атрибуты. Вложенность атрибутов не ограничена.

Для поля можно указывать имя атрибута, содержащего отображаемое значение поля CaptionSourceAttribute и имя атрибута по которому производится сортировка значений поля OrderSourceAttribute. Если эти атрибуты не указаны, то в качестве отображаемого значения и ключа сортировки используется значение самого поля.

Атрибут как и основное поле описывается объектом TfcxSourceField. Список атрибутов - TfcxSplitProperty.Attributes.

DateSplitPaths и TimeSplitPaths описывают какие составляющие частей даты и времени надо создавать для поля (в случае если это поле имеет тип дата или время).

Примеры:

```
var
  ARefField: TfcxReferenceAttributeSFProperties;
  AAttribute: TfcxSourceField;
begin
  //
  fcxDataSource1.AddFields;

  //
                                ,          ,          'Datel'
  fcxDataSource1.Fields.FieldName
['Datel'].SplitProperty.DateSplitPaths := [odt_Day, odt_Month,
odt_Year];
```


- **YAxisContainer** - ось Y, содержит поля с типом **TfcxAxisField**
- **PageContainer** - область фильтров, содержит поля с типом **TfcxAxisField**
- **MeasuresContainer** - показатели, содержит поля с типом **TfcxMeasureField**

Поля **TfcxAxisField** создаются на основе полей среза.

Показатели **TfcxMeasureField** могут быть созданы на основе полей среза, либо на основе скриптов **FastScript**.

Любое поле среза может быть перемещено в любой из контейнеров. Создание показателя на основе поля среза не запрещает этому полю находиться в другом контейнере.

2.3.1

Срез настраивается путем добавления полей среза в контейнеры.

Поле региона создается автоматически при добавлении в контейнер поля среза.

Для добавление измерения (поля) в регион оси или фильтров служат следующие методы контейнера:

function AddDimension(ASliceField: TfcxSliceField; AName: TfcxString = ""; ACaption: TfcxString = "): integer;

Функция добавляет в указанный регион измерение на основе поля ASliceField в конец списка полей. Возвращает позицию поля в списке полей региона.

Если поле региона на основе ASliceField уже существует, то оно переносится в указанный регион на нужную позицию с сохранением настроек, иначе поле региона создается.

procedure InsertDimension(ASliceField: TfcxSliceField; AIndex: integer; AName: TfcxString = ""; ACaption: TfcxString = ");

Метод вставляет в указанный регион измерение на основе поля ASliceField на заданную позицию в списке полей. Если поле региона на основе ASliceField уже существует, то оно переносится в указанный регион на нужную позицию с сохранением настроек, иначе поле региона создается.

procedure DeleteDimension(AIndex: integer);

Метод удаляет из указанного региона поле, находящееся на заданной позиции. Поле региона уничтожается.

Для управления уровнем показателей (поле "Показатели") служат методы контейнера:

function AddMeasuresField: integer;

Функция переносит поле "Показатели" в указанный регион, присваивая ему крайнюю позицию. Возвращает позицию поле "Показатели".

function InsertMeasuresField(AIndex: TfcxSmallCount): integer;

Функция переносит поле "Показатели" в указанный регион, присваивая ему указанную позицию. Возвращает позицию поле "Показатели".

procedure DeleteMeasuresField;

Функция удаляет поле "Показатели" из указанный региона. При этом поле "Показатели" автоматически переносится в регион фильтров на первую позицию.

ВАЖНО!

Поле "Показатели" является виртуальным полем, оно всегда создано и не присутствует в списке полей соответствующего контейнера.

Для обращения к его свойствам надо обращаться к свойству среза - **MeasuresContainer**.

Его положение в регионе определяется свойством **MeasuresContainer.Position**. Все поля региона, индекс которых равен или меньше **MeasuresContainer.Position**, отображаются в регионе после поля "Показатели".

Свойство **MeasuresContainer.Container** определяет контейнер, в котором находится поле "Показатели".

Операции по настройке структуры среза лучше объединять методами среза **BeginUpdate** и **EndUpdate**. Это позволит избежать лишних пересчетов и перестроений в срезе.

Примеры:

```
//
fcxSlice1.BeginUpdate;
//
//          Y          0
fcxSlice1.YAxisContainer.AddDimension(fcxSlice1.SliceField[0]);
//
//          Y          1          0
fcxSlice1.YAxisContainer.InsertDimension(fcxSlice1.SliceField[1], 0);
//
//          X          'FullName'
fcxSlice1.XAxisContainer.AddDimension(fcxSlice1.SliceFieldByName
['FullName']);
//
//          '          X
fcxSlice1.XAxisContainer.AddMeasuresField;
//
.
fcxSlice1.EndUpdate;
```

2.3.2

Показатели настраиваются путем добавления в контейнер **MeasuresContainer**.

Показатели создаются автоматически при добавлении в контейнер. При необходимости их можно создавать вручную и добавлять в контейнер созданный показатель.

Показатель создается на основе поля среза либо на основе скрипта **FastScript**.

Добавление показателя на основе ASliceField со статистической функцией AAgrFunc. Возвращает позицию показателя в контейнере:

```
function AddMeasure(ASliceField: TfcxSliceField; AName, ACaption: TfcxString;
AAgrFunc: TfcxAgrFunc): Integer;
```

Добавление вычисляемого показателя на основе скриптовой функции AScriptFunctionName с типом показателя AAgrFunc. AScriptFunctionCode - код функции. Возвращает позицию показателя в контейнере:

```
function AddCalcMeasure(AName, ACaption: TfcxString; AAgrFunc: TfcxAgrFunc;
AScriptFunctionName: String; AScriptFunctionCode: TfcxString): Integer;
```

Добавление созданного показателя AField. Возвращает позицию показателя в контейнере:

```
function AddMeasure(AField: TfcxMeasureField): Integer;
```

Вставка показателя на основе ASliceField со статистической функцией AAgrFunc на заданную позицию в контейнере:

```
procedure InsertMeasure(ASliceField: TfcxSliceField; AName, ACaption: TfcxString;
AAgrFunc: TfcxAgrFunc; AIndex: TfcxSmallCount);
```

Вставка вычисляемого показателя на основе скриптовой функции AScriptFunctionName с типом показателя AAgrFunc на заданную позицию в контейнере. AScriptFunctionCode - код функции:

```
procedure InsertCalcMeasure(AName, ACaption: TfcxString; AAgrFunc:
TfcxAgrFunc; AScriptFunctionName: String; AScriptFunctionCode: TfcxString; AIndex:
TfcxSmallCount);
```

Вставка созданного показателя на заданную позицию в контейнере:

```
procedure InsertMeasure(AField: TfcxMeasureField; AIndex: TfcxSmallCount);
```

Удаление показателя с заданным индексом

```
procedure DeleteMeasure(AMeasureIndex: TfcxSmallCount; ADoStopChange:
Boolean = False);
```

Для доступа к показателям и управления показателями необходимо пользоваться свойствами и методами **MeasuresContainer**.

Показатели могут быть скрыты. В этом случае они рассчитываются, но не отображаются.

Перемещение показателя в контейнере:

```
function MoveMeasure(AFromIndex, AToIndex: TfcxSmallCount): boolean;
```

Доступ к показателю:

```
property Measures[AIndex: TfcxSmallCount]: TfcxMeasureField;
```



```

//                                     12
'FirstName'
fcxSlice1.SliceFieldByName['FirstName'].UVFilterOf[12] := True;
//                                     (           )
fcxSlice1.SliceFieldByName['FirstName'].EndUpdateFieldFilter;

//                                     4
0
fcxSlice1.SliceField[0].UVSingleIndex := 4;

//                                     0
fcxSlice1.SliceField[0].InverseFilter;

//                                     ,
ARange
fcxSlice1.SliceField[0].SetRangeFilter(ARange);

//                                     "           "
SliceField[1].UVFilterType := uvft_Single;

```

2.5

Группы служат для повышения наглядности представления данных.

Управлять группами можно через методы и свойства поля среза и менеджера групп (**GroupManager**) поля среза.

Работать с группами (создавать, изменять и т.д.) можно после включения режима группировки в поле среза **CanGroup**.

Примеры:

```

//
ASliceField.CanGroup := True;
//
if ASliceField.CanGroup then
begin
  //                                     'Group1'
  AGroupIndex := ASliceField.GroupManager.CreateGroup('Group1').Index;
  //                                     3                                     AGroupIndex
  ASliceField.GroupManager.AddUVInGroup(3, AGroupIndex);
  //                                     30                                     AGroupIndex
  ASliceField.GroupManager.AddUVValueInGroup(30, AGroupIndex);
  //                                     "           "
  ASliceField.GroupManager.CreateOtherGroup;
end;

```

