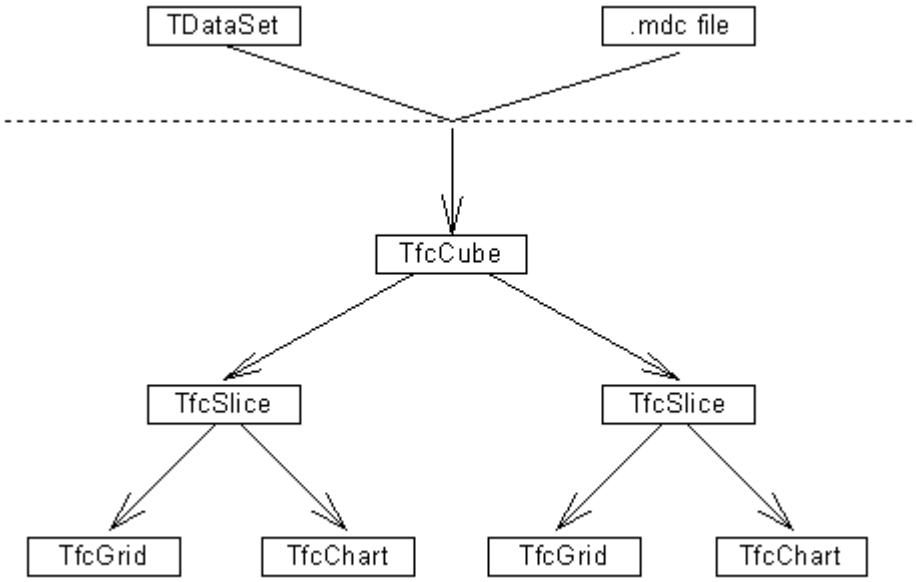
Fast Cube 1.0 Programmer manual

Table of Contents

Chapter 1

Fast Cube Architecture

The FastCube components library is a set of non-visual and visual components responsible for processing, storing and visualizing multidimensional data. The FastCube architecture is represented on the following chart:



A TDataSet descendant or a cube .mdc file can be used as a data source for the components.

The main non-visual components:
-TfcCube - (cube) the main data storage.
-TfcSlice - (slice) is used for storing the data layout and preparing the data in compliance with this structure.

The main visual components:
-TfcGrid - (grid) reflects the information according to data structure included in the slice and enables the user to manipulate the data and the structure.
-TfcToolBar - (grid toolbar) contains a set of buttons allowing to perform various operations with the grid, the slice and the cube.

Components for grid display:
-TfcChart (chart) is used for representing data in the form of a chart.
-TfcChartToolBar (chart toolbar) - contains a set of buttons allowing to perform various operations with the chart.

FastReport integration components:
-TfrcComponents (register) - an add-on component necessary for registering FastCube in the FastReport designer.

-TfcGridReport (grid report) - the component that allows the user to print the grid with Fast Report without using a report designer first.

-TfrcGrid - to link TfcGrid with TfrcCrossView

-TfrcChart - to link TfcChart with TfrcChartView

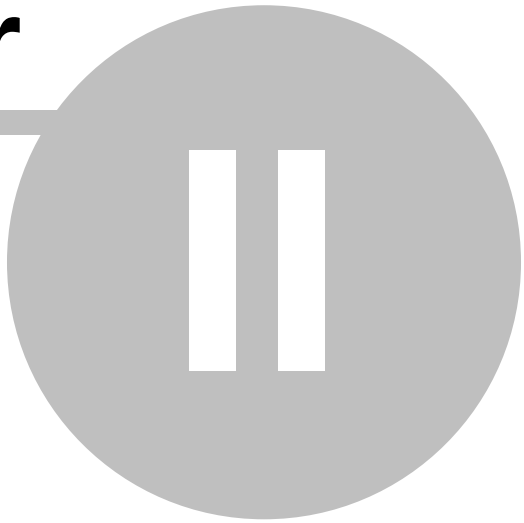
Components used in the FastReport designer:

-TfrcCube (cube view) is the main data storage for creating reports.

-TfrcCrossView (cross view) is used for setting the slice properties and printing cross grid.

-TfrcChartView - (cross chart) is used for setting the chart properties and printing chart.

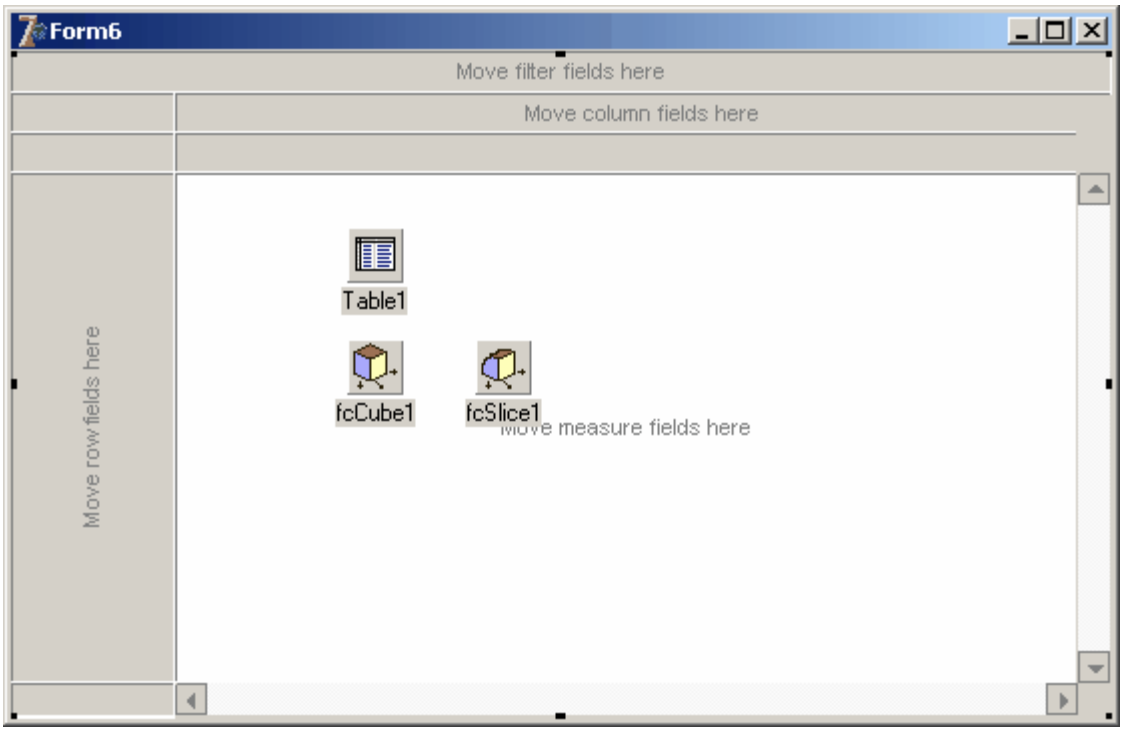
Chapter



Example of Use

1. In order to create a simple attachment displaying data from some database table place the following components on an empty form:

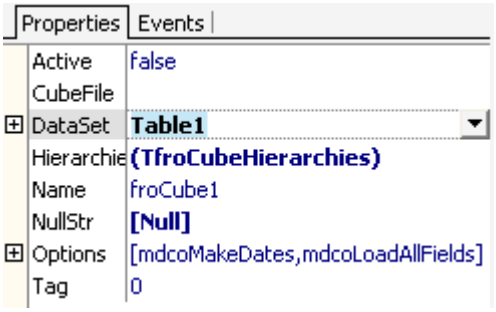
– Table1: TTable – to get a data set from the table through BDE.
– fcCube1: TfcCube – cube – a multidimensional data source.
– fcSlice1: TfcSlice – a selection of cube data.
– fcGrid1: TfcGrid – grid – is a visual component representing multidimensional data.



2. Connect TTable with a certain table. For example, the Country table from the DBDemos database:

– Set DatabaseName = DBDemos
– TableType = ttParadox
– TableName = country

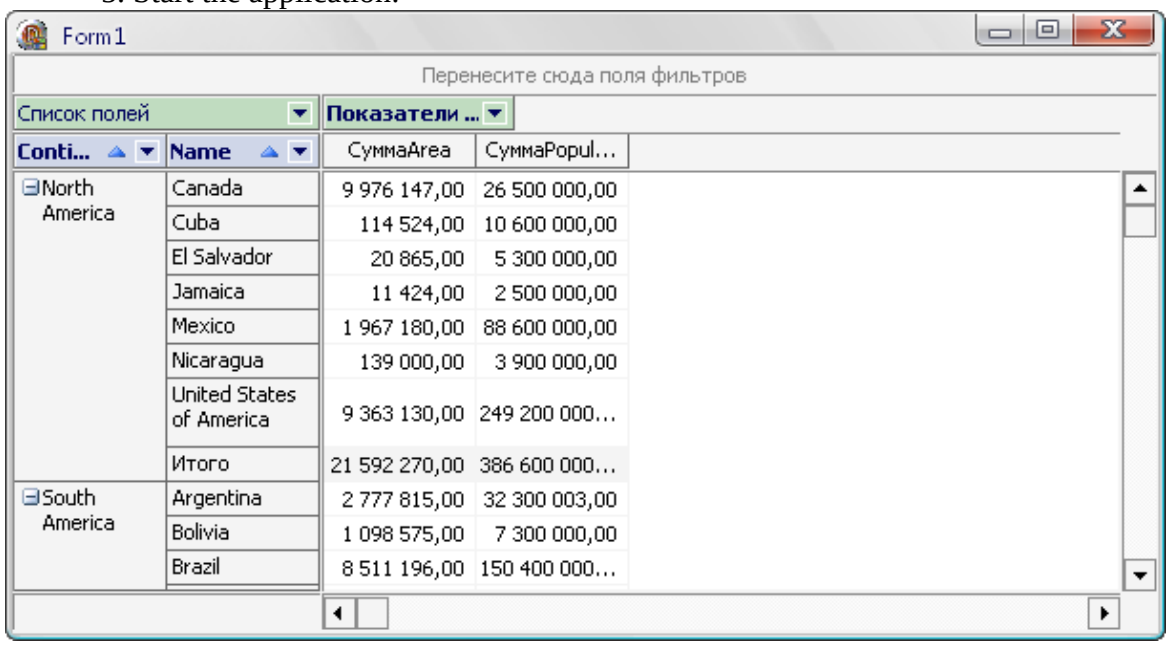
3. Link fcCube1 with Table1:



4. Define the grid layout in the shape constructor:

```
constructor TForm1.Create(AOwner: TComponent);
begin
  inherited;
  fcCube1.Active := True;
  // fill Y Axis
  fcSlice1.AddFieldTo('Continent', 'Continent', rf_CapYAx);
  fcSlice1.AddFieldTo('Name', 'Name', rf_CapYAx);

  // fill facts
  fcSlice1.AddFieldTo('Area', 'Area', rf_CapFacts, af_Sum);
  fcSlice1.AddFieldTo('Population', 'Population', rf_CapFacts, af_Sum);

  // Add Measures to X Axis
  fcSlice1.AddFieldTo(sMeasuresFieldName, '', rf_CapXAx);
end;
```

5. Start the application:

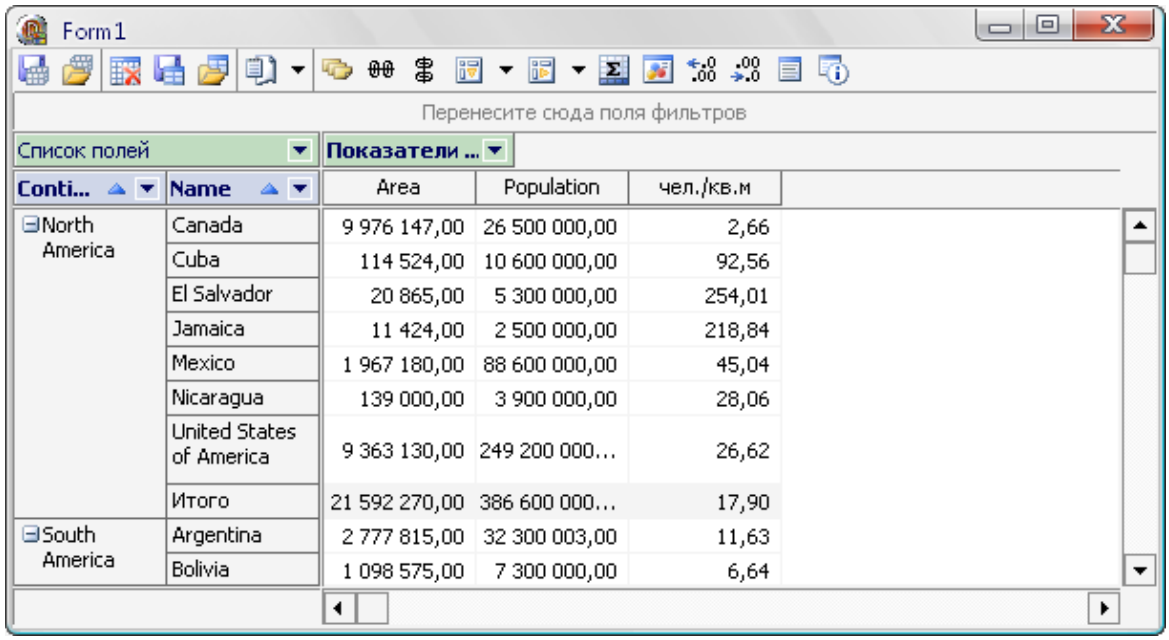


6. Then add the calculated fact and the calculation formula Population / Area, and then change the code in such a way that all the settings are executed by one action:

```
constructor TForm1.Create(AOwner: TComponent);
var
  FieldIndex: Integer;
begin
  inherited;
  fcCube1.Active := True;

  // start operations
  fcSlice1.BeginUpdate;

  // fill Y Axis
  fcSlice1.AddFieldTo('Continent', 'Continent', rf_CapYAx);
  fcSlice1.AddFieldTo('Name', 'Name', rf_CapYAx);

  // fill facts
  fcSlice1.AddFieldTo('Area', 'Area', rf_CapFacts, af_Sum);
  fcSlice1.AddFieldTo('Population', 'Population', rf_CapFacts, af_Sum);

  // add calculated fact Population / Area
  FieldIndex := fcSlice1.AddCalcFieldTo(
    'begin '#$D#$A+
    '  if <Area> <> 0 then'#$D#$A+
    '    Result := <Population> / <Area> else'#$D#$A+
    '    Result := 0;'#$D#$A+
    'end.', 'person./sq.m.', 1, af_Formula);

  // Add Measures to X Axis
  fcSlice1.AddFieldTo(sMeasuresFieldName, '', rf_CapXAx);

  // finish operation
  fcSlice1.EndUpdate;
end;
```

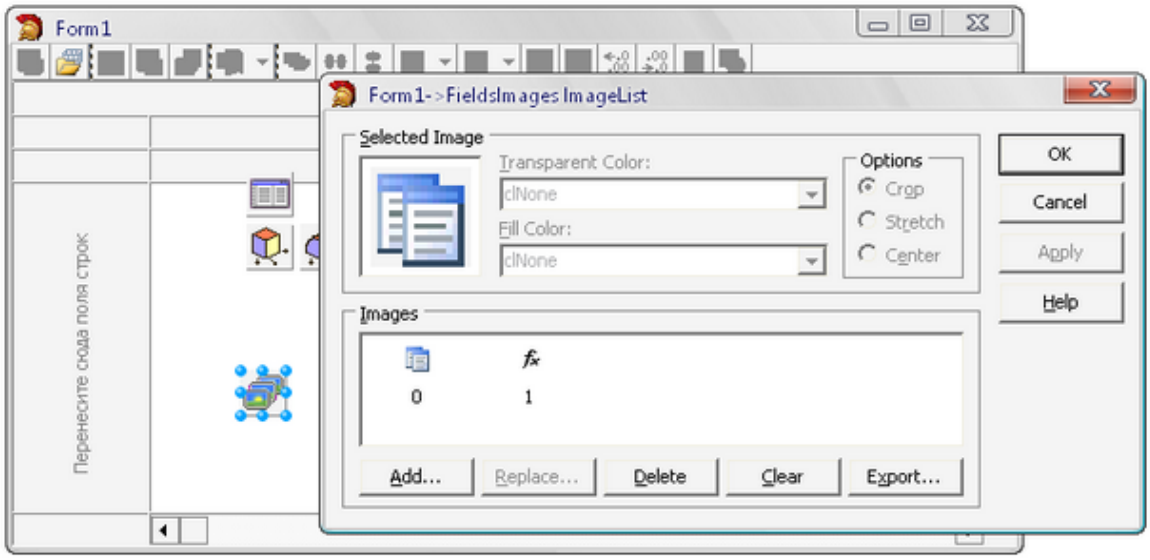
Please pay attention to the functions BeginUpdate and EndUpdate – they make it unnecessary to update the axes and facts upon the completion of each operation. The addition of the calculated fact in the code is performed by running the AddCalFieldTo function, which returns the index of the fact on the list of facts.

7. Add a grid control panel to the form and connect it with the grid through grid options. Then restart the application once again. As a result you should get the following window:



8. To make your application a little more colourful, add the FieldImages component to the form:

TImageList and fill it up with pictures for the field list and the “Index” window:



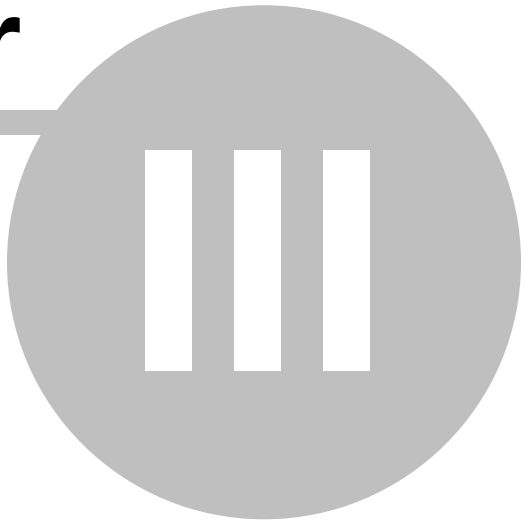
Set the FieldImages component in the FieldImages settings of the fcGrid1 component:

```
procedure TForm1.fcGrid1GetFieldImageIndex(Sender: TObject; Region:
TfcTypeDrRegion; Field: TfcFieldOfRegion; var ImageIndex: Integer);
begin
  if Region = dr_Tool then
    ImageIndex := 0 else
```

```
  if Field = fcGrid1.Slice.MeasureField then
    ImageIndex := 1 else
    ImageIndex := -1;
end;
```

If while you are doing all this something doesn’t work, you can open the demo example from the Demos\Simple catalog and look at how this was performed in it.

Chapter



The Cube

TfcCube - (cube) the main data storage.

TfcCube is a non-visual component responsible for storing and front-end data processing. A TDataSet descendant or a cube .mdc file with a special data storage format can be used as a data source for the cube. A cube is a data source for all the other FastCube components.

```
 TfcCube = class(TComponent)
  public
   constructor Create(AOwner: TComponent); override;
   destructor Destroy; override;
   procedure Open;
   procedure Close;
   procedure SaveToStream(Stream: TStream);
   function LoadFromStream(Stream: TStream): Boolean;
   procedure SaveToFile(AFileName: string);
   procedure LoadFromFile(AFileName: string);
   procedure AddField(AFieldName, AFieldCaption: string);
   procedure SetBaseField(AField, ABaseField: string);
   function HaveBaseField(AField: string): Boolean;
   procedure ClearBaseFields;
   function FindUnique(AFieldIndex: integer; Value: Variant; var AIndex: integer):
Boolean;
   procedure BeginUpdate;
   procedure EndUpdate;
   procedure ClearConvert;
   procedure SaveConvertToStream(Stream: TStream);
   procedure SaveConvertToFile(AFileName: String);
   function LoadConverTfcmStream(Stream: TStream): Boolean;
   procedure LoadConverTfcmFile(AFileName: String);
   procedure Refresh;
   procedure DesignOpen;

   property UniqueValue[AFieldIndex, AValueIndex: integer]: Variant read
GetUniqueValue;
   property UniqueCount[AFieldIndex: integer]: Integer read GetUniqueCount;
   property SourceCount: Integer read GetSourceCount;
   property UniqueValueAtRow[ARow, AFieldIndex: integer]: variant read
GetUniqueValueAtRow;
   property UniqueCaption[AFieldIndex, AValueIndex: integer]: string read
GetUniqueCaption;
   property UniqueCaptionAtRow[ARow, AFieldIndex: integer]: string read
GetUniqueCaptionAtRow;
```

```
    property SourceRecs[ARow: integer]: PUniqueArray read GetSourceRec;
    property DisplayFormat[AFieldIndex: integer]: TfcFormat read GetDisplayFormat;
    property AsVariant[ADataRow: PUniqueArray; AFieldNo: Integer]: Variant read
GetAsVariant;
    property Slices: TfcSlices read FSlices;
    property FieldCount: Integer read FFieldCount;
    property CubeFields: TfcCubeFields read FCubeFields;
    property CubeStream: TStream read FCubeStream write FCubeStream;
    property Collecting: boolean read FCollecting write SetCollecting;
    property Caption: string read FCaption write SetCaption;
    property Description: string read FDescription write SetDescription;
  published
    property DataSet: TDataSet read FDataSet write SetDataSet;
    property CubeFile: string read FCubeFile write SetCubeFile;
    property Options: TfcCubeOptions read FOptions write SetOptions default
[mdcoMakeDates, mdcoLoadAllFields];
    property NullStr: string read FNullStr write FNullStr;
    property Active: Boolean read FActive write SetActive default False;
    property DefaultFormat: TfcDefaultFormat read FDefaultFormat write
SetDefaultFormat;
    property UseMultiLoad: Boolean read FUseMultiLoad write SetUseMultiLoad;
    property OnGetNextDataset: TfcGetNextDatasetEvent read FOnGetNextDataset
write FOnGetNextDataset;
    property OnGetFieldConv: TfcCubeGetFieldConvEvent read FOnGetFieldConv write
FOnGetFieldConv;

    property OnCubeChanged: TNotifyEvent read FOnCubeChanged write
FOnCubeChanged;
    property OnNeedSlice: TfcCubeOnNeedSlice read FOnNeedSlice write
FOnNeedSlice;
    property BeforeOpen: TNotifyEvent read FBeforeOpen write FBeforeOpen;
    property AfterOpen: TNotifyEvent read FAfterOpen write FAfterOpen;
    property BeforeClose: TNotifyEvent read FBeforeClose write FBeforeClose;
    property AfterClose: TNotifyEvent read FAfterClose write FAfterClose;
  end;
```

The TfcCube class includes the following methods:

- *Open* – fill up the cube's structures with the data from a set source. If TDataset is the data source, and it is closed, it is opened, and upon the completion of the upload returned to the original status.
- *Close* – clean up the internal structures of the cube and makes the cube inactive.
- *SaveToStream* – save the cube contents into stream.
- *LoadFromStream* – load the cube from stream.
- *SaveToFile* – save the cube to file.
- *LoadFromFile* – load the cube from file.

- *AddField* – add the field to the list of uploaded fields and sets the field caption. If the **mdcoLoadAllFields** setting is on, it is used for setting the caption.

- *SetBaseField* – set the base field for another data sample field. The base field will be used as the real field value (it will be used in data storage and grouping), and the field for which the base field was set will be only used for uploading the values.

- *HaveBaseField* – a check-up of whether the base field was set for the transferred field.

- *ClearBaseFields* – clear the base fields settings.

- *FindUnique* – search for a unique Value for a field with the AFieldIndex index on the list of unique values. Returns the search results. The index of the element found in the search is placed in the AIndex variable.

- *BeginUpdate* – start updating the structure. No concurrent operations, like recalculation of axes or facts and updating listeners, will be run until the changes are fixed.

- *EndUpdate* – save changes in the structure.

- *ClearConvert* - clear description of field converter;

- *SaveConvertToStream* - save description of field converter to stream;

- *SaveConvertToFile* - save description of field converter to file;

- *LoadConverTfcmStream* - load description of field converter from stream;

- *LoadConverTfcmFile* - load description of field converter from file;

- *Refresh* - Refresh data in cube;

The following properties are defined in the TfcCube class:

- *DataSet* – a data set (one of the possible data sources for a cube).

- *CubeFile* – the name of the external cube file (one of the possible data sources for a cube).

- *CubeStream* – a link to a stream from which the cube can be uploaded (one of the possible data sources for a cube).

- *Active* – shows if the cube is active and calls Open or Close depending on the cube's state.

- *NullStr* – the row that will be inserted from the cube instead of the Null variable. *'[Null]'* by default.

- *Options* include the following cube properties:

  - **mdcoMakeDates** – when a field of the DATE type is uploaded, allows to automatically upload of the corresponding fields: quarter, month, week, day.

  - **mdcoLoadAllFields** – upload all available fields from a data source. If this option is not set, only the fields set earlier get uploaded. The list of fields for uploading is filled out by means of running the **AddField** method.

  - **mdcoRefreshOnDatasetReopen** – in case of reiterated opening of the data source that was not initiated by the cube, the data in the cube is refreshed.

  - **mdcoShowConvertor -** Show editor to set field type converting properties

  - **mdcoMakeTimes -** when a field of the TIME type is uploaded, allows to

automatically upload of the corresponding fields: Hour, Minute and Second.

- **mdcoLoadWithDefaultFormat -** use DefaultFormat on Load from cube
- **mdcoUseYearInWeekOfYear -** use year in week of year

By default the **mdcoMakeDates** and **mdcoLoadAllFields** properties are on.

- *Caption* – the cube caption. May also be displayed in a grid.
- *Description* – the description of the cube.
- *Collecting* – the state of data reading.
- *Slices* – a collection of slices (TfcSlice) which retrieve the data from the cube.
- *FieldCount* – the number of fields in the cube.
- *CubeFields* – the fields of the cube.
- *UniqueCount* - the number of unique values for a field.

UniqueCount – an array of unique values for a field.

UniqueCaption – an array of captions of unique values for a field.

- *SourceCount* - the number of source entries on which the cube was based.
- *UniqueValueAtRow* - the unique value for the field in the set row of source entries.
- *UniqueValueAtRow* - the unique value for the field in the set row of source entries.
- *SourceRecs* - the array of source entries.
- *AsVariant* - getting the value of the source entry in the form of a Variant.
- *DisplayFormat* - format of field;
- *UseMultiLoad* - load data from from multiple sources with the same structure. Need event OnGetNextDataset.
- *DefaultFormat* - default formats for fields.

The following events are defined in the TfcCube class:

- OnCubeChanged is called when the cube is altered (opened, uploaded).
- OnNeedSlice - a query for creating slices during upload of a previously saved cube. Displays the number of slices necessary for start up. This event should be processed in the cases when the number of slices referring to the cube is smaller than the corresponding number displayed when the cube is saved. The matter is that one cube used as a data source may be linked with several independent slices, in which the information is organized in different ways. And when the cube is saved, all the schemes connected with it are saved, too. This is why while the cube is being uploaded, it can fill out the same number of slices with schemes. If the number of slices shown after this event is called is smaller, only part of the schemes saved in the cube will be uploaded.
- BeforeOpen is called before running the Open command.
- AfterOpen is called after running the Open command.
- BeforeClose is called before running the Close command.
- AfterClose is called after running the Close command.

- OnGetNextDataset - called to request the following dataset (with UseMultiLoad).
- OnGetFieldConv - set properties to convert the type of source fields.

Chapter IV

The Slice

TfcSlice - (slice) is used for storing the data layout and bringing the data in compliance with this structure. A slice is an intermediate component between the cube and the components responsible for information visualization. It stores the structure of data visualization (scheme), filters, and performance indicators.

```
TfcSlice = class(TfcAbstractSlice)
  public
    constructor Create(AOwner : TComponent); override;
    destructor Destroy; override;
    procedure BeginUpdate;
    procedure EndUpdate;
    procedure Clear;
    // add/remove slice event listner
    procedure AddToUpdates(Obj : TObject);
    procedure RemoveFromUpdates(Obj : TObject);
    procedure ManageFacts(AField : TfcSliceField);
    procedure RangeConfig(DefaultColor: LongInt; FieldIndex: Integer = -1);
    procedure ChangeDecimal(FieldIndex: Integer; Increment: Integer);
    procedure FillRangesList(List: TList);
    procedure GetValueColor(Value: Variant; FactIndex: Integer; var TextColor,
BackColor: LongInt);
    procedure ShowMetrix;
    function HaveLayout: Boolean;
    function IsMeasureField(ARegion : TfcRegionOfField; AIndex : Integer) : Boolean;
    // slice read/write
    procedure SaveToStream(Stream : TStream); override;
    procedure SaveToFile(AFileName: String);
    procedure LoadFromStream(Stream : TStream); override;
    procedure LoadFromFile(AFileName: String);
    // uniques filter
    procedure SetAllFilter(ARegion : TfcRegionOfField; AIndex : Integer);
    procedure InverseFilter(ARegion : TfcRegionOfField; AIndex: integer);
    procedure SetNoneFilter(ARegion : TfcRegionOfField; AIndex: integer);
    // fact work
    function GetFactFields(Field: TfcSliceField) : TStringList;
    procedure SetFactFields(List: TStringList);
    procedure AddFactFields(List: TStringList);
    // drill trough
    function GetDetailInfoAbs(XAxisIndex, YAxisIndex : Integer) : TfcDetailInfo;
    function GetDetailInfoVis(XAxisIndex, YAxisIndex : Integer) : TfcDetailInfo;
    // work with adding/moving/removing fields from one region to another
```

```
    function AddFieldTo(AFieldName: String; ACaption: String; ARegionOfField:
TfcRegionOfField; AAgrFunc: TfcAgrFunc = af_None; AFilterText: String = ''): integer;
overload;
    function AddFieldTo(AName: String; AFieldName: String; ACaption: String;
ARegionOfField: TfcRegionOfField; AAgrFunc: TfcAgrFunc = af_None; AFilterText:
String = ''): integer; overload;
    function AddCalcFieldTo(AFormula: String; ACaption: String; AOrder: Integer;
AAgrFunc: TfcAgrFunc; AFilterText: String = ''): integer; overload;
    function AddCalcFieldTo(AName: String; AFormula: String; ACaption: String;
AOrder: Integer; AAgrFunc: TfcAgrFunc; AFilterText: String = ''): integer; overload;
    function InsertFieldTo(Index: Integer; AFieldName: String; ACaption: String;
ARegionOfField: TfcRegionOfField; AAgrFunc: TfcAgrFunc = af_None; AFilterText:
String = ''): integer; overload;
    function InsertFieldTo(Index: Integer; AName: String; AFieldName: String; ACaption:
String; ARegionOfField: TfcRegionOfField; AAgrFunc: TfcAgrFunc = af_None;
AFilterText: String = ''): integer; overload;
    procedure RemoveFieldFrom(AFieldName: String; ACaption: String;
ARegionOfField: TfcRegionOfField; AAgrFunc: TfcAgrFunc = af_None); overload;
    procedure RemoveFieldFrom(AName: String; ARegionOfField: TfcRegionOfField);
overload;
    procedure MoveBefore(AFieldIndex, AFieldIndexBefore: Integer; ARegionOfField:
TfcRegionOfField);
    // add/remove special measure - system counter
    function AddCountToFact:integer;
    procedure RemoveCounTfcmFact;
    function FieldsOfRegion(ARegionOfField: TfcRegionOfField): TfcFieldsOfRegion;
    // axis tree walkers
    function TraverseAxis(ARowAxis: Boolean; AFirstItem, ALastItem, AFirstDrawed:
Integer; Proc: TfcSliceAxisTraverseProc): TfcSliceDrawHeaderResult; overload;
    function TraverseAxis(ARowAxis: Boolean; AFirstItem, ALastItem, AFirstDrawed:
Integer; Proc: TfcSliceAxisTraverseProc; AGetColWidth : TfcSliceAxisGetColWidth;
AGetRowHeight : TfcSliceAxisGetRowHeight): TfcSliceDrawHeaderResult; overload;
    // properties
    property SliceFields : TfcSliceFields read FSliceFields;
    property StoredSchema: string read FStoredSchema;
    property BuildingData : boolean read FBuildingData;
    property Active : Boolean read GetActive;
    property HideRowZeros: Boolean read FHideRowZeros write SetHideRowZeros;
    property HideColZeros: Boolean read FHideColZeros write SetHideColZeros;
    property TypeSortOfXAxis: TfcTypeSortAxis read GetTypeSortOfXAxis write
SetTypeSortOfXAxis;
    property TypeSortOfYAxis: TfcTypeSortAxis read GetTypeSortOfYAxis write
SetTypeSortOfYAxis;
    property SelectedFact: Integer read FSelectedFact write SetSelectedFact;
    property SelectedCol : Integer read FSelectedCol  write SetSelectedCol;
    property SelectedRow : Integer read FSelectedRow  write SetSelectedRow;
```

```
  property RegionFieldsCount[ARegion: TfcRegionOfField]: Integer read
GetRegionFieldsCount;
  property RegionFieldCaption[ARegion: TfcRegionOfField; AFieldIndex: Integer]:
string read GetRegionFieldCaption;
  property RegionFieldFilter[ARegion: TfcRegionOfField; AFieldIndex: Integer]:
Boolean read GetRegionFieldFilter;
  property RegionFieldExpanded[ARegion : TfcRegionOfField; AFieldIndex: Integer] :
Boolean read GetRegionFieldExpanded write SetRegionFieldExpanded;
  property XAxisValueAbsIndexInLevel[ATargetLevel, AVisIndex: Integer]: Integer
    read GetXAxisValueAbsIndexInLevel;
  property YAxisValueAbsIndexInLevel[ATargetLevel, AVisIndex: Integer]: Integer
    read GetYAxisValueAbsIndexInLevel;
  property XAxisValueExpandedAbs[ALevel, AIndex: integer]: Boolean read
    GetXAxisValueExpandedAbs write SetXAxisValueExpandedAbs;
  property YAxisValueExpandedAbs[ALevel, AIndex: integer]: Boolean read
    GetYAxisValueExpandedAbs write SetYAxisValueExpandedAbs;
  property XAxisValueExpandedVis[ALevel, AVisIndex: integer]: Boolean read
    GetXAxisValueExpandedVis write SetXAxisValueExpandedVis;
  property YAxisValueExpandedVis[ALevel, AVisIndex: integer]: Boolean read
    GetYAxisValueExpandedVis write SetYAxisValueExpandedVis;
  property XAxisValueMeasureValueVis[AVisIndex: integer]: Integer read
    GetXAxisValueMeasureValueVis;
  property YAxisValueMeasureValueVis[AVisIndex: integer]: Integer read
    GetYAxisValueMeasureValueVis;
  property XAxisFullExpanded[ALevel: integer]: Boolean read
GetXAxisFullExpanded write SetXAxisFullExpanded;
  property YAxisFullExpanded[ALevel: integer]: Boolean read
GetYAxisFullExpanded write SetYAxisFullExpanded;
  property XAxisValueVisibleChildCountAbs[ALevel, AIndex: Integer]: Integer read
    GetXAxisValueVisibleChildCountAbs;
  property YAxisValueVisibleChildCountAbs[ALevel, AIndex: Integer]: Integer read
    GetYAxisValueVisibleChildCountAbs;
  property XAxisAllCountInLevel[ALevel: Integer]: Integer read
    GetXAxisAllCountInLevel;
  property YAxisAllCountInLevel[ALevel: Integer]: Integer read
    GetYAxisAllCountInLevel;
  property XAxisVisibleItemCount: Integer read GetXAxisVisibleItemCount;
  property YAxisVisibleItemCount: Integer read GetYAxisVisibleItemCount;
  property XAxisVisibleItemCountInLevel[ALevel: Integer]: Integer read
    GetXAxisVisibleItemCountInLevel;
  property YAxisVisibleItemCountInLevel[ALevel: Integer]: Integer read
    GetYAxisVisibleItemCountInLevel;
  property XAxisValueChildCountAbs[ALevel, AIndex: Integer]: Integer read
    GetXAxisValueChildCountAbs;
  property YAxisValueChildCountAbs[ALevel, AIndex: Integer]: Integer read
```

```
    GetYAxisValueChildCountAbs;
  property XAxisFieldOrder[Index : Integer]: Boolean read GetXAxisFieldOrder
    write SetXAxisFieldOrder;
  property YAxisFieldOrder[Index : Integer]: Boolean read GetYAxisFieldOrder
    write SetYAxisFieldOrder;
  property XAxisExistsVisibleInLevel[ALevel: Integer]: Boolean read
    GetXAxisExistsVisibleInLevel;
  property YAxisExistsVisibleInLevel[ALevel: Integer]: Boolean read
    GetYAxisExistsVisibleInLevel;
  property XAxisValueTypeAbs[ALevel, AIndex: Integer]: TfcTypeOfCellAxis read
    GetXAxisValueTypeAbs;
  property YAxisValueTypeAbs[ALevel, AIndex: Integer]: TfcTypeOfCellAxis read
    GetYAxisValueTypeAbs;
  property XAxisValueTypeVis[ALevel, AVisIndex: integer]: TfcTypeOfCellAxis read
    GetXAxisValueTypeVis;
  property YAxisValueTypeVis[ALevel, AVisIndex: integer]: TfcTypeOfCellAxis read
    GetYAxisValueTypeVis;
  property XAxisValueParentIndexVis[ALevel, AVisIndex: integer]: Integer read
    GetXAxisValueParentIndexVis;
  property YAxisValueParentIndexVis[ALevel, AVisIndex: integer]: Integer read
    GetYAxisValueParentIndexVis;
  // Check State of Cell in Axis (Any State Included)
  property XAxisIncludeStateAnyAbs[ALevel, AIndex: Integer; AState: byte]:
    boolean read GetXAxisIncludeStateAnyAbs;
  property YAxisIncludeStateAnyAbs[ALevel, AIndex: Integer; AState: byte]:
    boolean read GetYAxisIncludeStateAnyAbs;
  property XAxisIncludeStateAnyVis[ALevel, AVisIndex: Integer; AState: byte]:
    boolean read GetXAxisIncludeStateAnyVis;
  property YAxisIncludeStateAnyVis[ALevel, AVisIndex: Integer; AState: byte]:
    boolean read GetYAxisIncludeStateAnyVis;
  // Check State of Cell in Axis (All State Included)
  property XAxisIncludeStateAllAbs[ALevel, AIndex: Integer; AState: byte]:
    boolean read GetXAxisIncludeStateAllAbs;
  property YAxisIncludeStateAllAbs[ALevel, AIndex: Integer; AState: byte]:
    boolean read GetYAxisIncludeStateAllAbs;
  property XAxisIncludeStateAllVis[ALevel, AVisIndex: Integer; AState: byte]:
    boolean read GetXAxisIncludeStateAllVis;
  property YAxisIncludeStateAllVis[ALevel, AVisIndex: Integer; AState: byte]:
    boolean read GetYAxisIncludeStateAllVis;
  property XAxisValueVis[ALevel, AIndex: Integer]: String read GetXAxisValueVis;
  property XAxisValueAbs[ALevel, AIndex: Integer]: String read GetXAxisValueAbs;
  property YAxisValueVis[ALevel, AIndex: Integer]: String read GetYAxisValueVis;
  property YAxisValueAbs[ALevel, AIndex: Integer]: String read GetYAxisValueAbs;
  // width of columns
  property XAxisColWidth[ACol : Integer]: Integer read GetXAxisColWidth write
SetXAxisColWidth;
```

```
    property YAxisColWidth[ACol : Integer]: Integer read GetYAxisColWidth write
SetYAxisColWidth;
    // Height of rows like width of cols
    property YAxisRowHeight[ARow : Integer]: Integer read GetYAxisRowHeight write
SetYAxisRowHeight;
    property XAxisRowHeight[ARow : Integer]: Integer read GetXAxisRowHeight write
SetXAxisRowHeight;
    property MeasureIndex[ACol, ARow: Integer]: Integer read GetMeasureIndex;
    // Data value as Variant
    property DataVariantVis[AVisCol, AVisRow: Integer]: Variant read
GetDataVariantVis;
    property DataVariantAbs[ACol, ARow: Integer]: Variant read GetDataVariantAbs;
    // Data value as formatted string
    property DataStringVis[AVisCol, AVisRow:Integer]: String read GetDataStringVis;
    property DataStringAbs[ACol,ARow:Integer]: String read GetDataStringAbs;
    // Data value as percent
    property DataPercentStringVis[AVisCol, AVisRow: Integer; DisplayAs:
TfcDisplayAs]: String read GetDataPercentStringVis;
    property DataPercentVariantVis[ACol,ARow:Integer; DisplayAs: TfcDisplayAs]:
Variant read GetDataPercentVariantVis;
    property DataAlignmentAbs[AIndex: Integer]: TAlignment read
GetDataAlignmentAbs write SetDataAlignmentAbs;
    property DataAlignmentVis[AVisIndex: Integer]: TAlignment read
GetDataAlignmentVis write SetDataAlignmentVis;
    property CapXFields[AIndex: integer]: TfcFieldOfRegion read GetCapXFields;
    property CapYFields[AIndex: integer]: TfcFieldOfRegion read GetCapYFields;
    property CapPageFields[AIndex: integer]: TfcFieldOfRegion read GetCapPageFields;
    property CapFactsFieldsAbs[AIndex: integer]: TfcFieldOfRegion read
      GetCapFactsFieldsAbs;
    property CapFactsFieldsVis[AVisIndex: Integer]: TfcFieldOfRegion read
      GetCapFactsFieldsVis;
    property CapFactsFieldIndexVis[AVisIndex: Integer]: Integer read
      GetCapFactsFieldIndexVis;
    property VisibleCapFactsFieldsCount: Integer read FVisibleCapFactsFieldsCount;
    property YAxisLevelCount : Integer read GetYAxisLevelCount;
    property XAxisLevelCount : Integer read GetXAxisLevelCount;
    property YAxisVisibleLevelCount: Integer read GetYAxisVisibleLevelCount;
    property XAxisVisibleLevelCount: Integer read GetXAxisVisibleLevelCount;
    property UniqueCount[ARegion : TfcRegionOfField; AIndex: integer]: Integer read
GetUniqueCount;
    property UniqueFilter[ARegion : TfcRegionOfField; AIndex, AUniqueIndex: integer]:
Boolean read GetUniqueFilter write SetUniqueFilter;
    property UniqueCaption[ARegion : TfcRegionOfField; AIndex, AUniqueIndex:
integer]: String read GetUniqueCaption;
    property TreeFilter[ARegion : TfcRegionOfField; AIndex, AHierLevel,
      AUniqueIndex: integer]: Boolean read GetTreeFilter write SetTreeFilter;
```

```
    property SliceUniques: TfcSliceUniques read FSliceUniques;
    property FieldCount: integer read GetFieldCount;
    property MeasureIndexInY: integer read GetMeasureIndexInY;
    property MeasureIndexInX: integer read GetMeasureIndexInX;
    property ActiveMeasure: Boolean read GetActiveMeasure;
    property MeasureField: TfcFieldOfRegion read FMeasureField;
    property Locked : Boolean read FLocked;
    property XAxis : TfcAxis read FValuesOfXAxis;
    property YAxis : TfcAxis read FValuesOfYAxis;
    property FilterRecCount: integer read GetFilterRecCount;
    property Filters: TfcSliceFilterManager read FFilters;
    property SourceRec[ARecIndex: integer]: PUniqueArray read GetSourceRec;
published
    property DefaultColWidth:Integer read FDefaultColWidth write SetDefaultColWidth
default 80;
    property DefaultRowHeight: Integer read FDefaultRowHeight write
SetDefaultRowHeight default 18;

    property Options : TfcSliceOptions read FOptions write SetOptions;
    property Cube  : TfcCube read FCube write SetCube;
    property NullStr : String read FNullStr write SetNullStr;
    property UnAssignedStr : String read FUnAssignedStr write SetUnAssignedStr;

    property PopUpFieldListWidth: integer read FPopUpFieldListWidth write
FPopUpFieldListWidth default 120;
    property PopUpWidthDefault: integer read FPopUpWidthDefault write
FPopUpWidthDefault default 120;
    property FieldsOrder: TfcFieldsOrder read FFieldsOrder write SetFieldsOrder;
    property ShowSplitFieldsInFieldList: TfcShowSplitFields read FShowSplitFieldsInFieldList write
    property StoreInDFM: Boolean read FStoreInDFM write FStoreInDFM default False;

    property OnFieldsListSortCompare: TfcFieldsListSortCompare read
FOnFieldsListSortCompare write FOnFieldsListSortCompare;
    property OnInterpreterCreated: TfcSliceInterpreterCreated read
FOnInterpreterCreated write FOnInterpreterCreated;
    property OnStartChange: TfcSliceStartChange read FOnStartChange write
FOnStartChange;
    property OnStopChange: TfcSliceStopChange read FOnStopChange write
FOnStopChange;
    property OnSliceChanged: TfcSliceChanged read FOnSliceChanged write
FOnSliceChanged;
    property OnBeforeRemoveSliceFieldFromRegion: TfcSliceFieldRegionChange read
FOnBeforeRemoveSliceField write FOnBeforeRemoveSliceField;
    property OnBeforeAddedSliceFieldToRegion: TfcSliceFieldRegionChange read
FOnBeforeAddedSliceFieldToRegion write FOnBeforeAddedSliceFieldToRegion;
```

property OnAfterAddedSliceFieldToRegion: TfcSliceFieldRegionChanged **read** FOnAfterAddedSliceFieldToRegion **write** FOnAfterAddedSliceFieldToRegion;
end;

While using many of the slice's methods and properties, one needs to bear in mind the coordinate system of its axes. It is different from the one displayed on the screen and in the grid. Only the visible cells of the axes are displayed in the grid, while all the cells are stored in the slice. Each cell contains information on its status, including the information on whether the cell is visible or not.

The visible coordinates, or the grid coordinates differ from the absolute coordinates: they do not take into account the invisible cells in the axes. Such cells include hidden zeros, folded (grouped) regions and other cells used for optimizing the internal slice.

In case there is a method or a property used both for absolute coordinates and for visible coordinates, the difference between them must be obvious in the postfix. The Abs postfix means that absolute coordinates are expected in the method as parameters, and the Vis postfix, on the contrary, refers to visible coordinates.

The TfcSlice class includes the following methods:

- *BeginUpdate* – start updating the structure. No concurrent operations, like recalculation of axes or facts and updating listeners, will be run until the changes are fixed.
- *EndUpdate* – save changes in the structure.
- *Clear* - clear the structure (delete all changes/indices, clear filters).
- *AddToUpdates* - add a listener Obj to the list of listeners. For a listener, the ExecuteAction function with the TfcSliceAction parameter is called if the slice is being changed.
- *RemoveFromUpdates* - remove a listener Obj from the list of listeners.
- *ManageFacts* - call the fact settings dialog.
- *FillRangesList* – fill out the list of data capturing ranges.
- *RangeConfig* - call the range settings dialog. If FieldIndex <> -1 the FieldIndex list for the fact will automatically open in the dialogue.
- *GetValueColor* - get the text and background colours for the Value of the FactIndex fact.
- *ChangeDecimal* - change the precision of displaying the FieldIndex fact by the Increment value. In case the Increment value is negative, the precision will be decreased.
- *SaveToStream* – save scheme to stream.
- *SaveToFile* – saves the scheme to file.
- *LoadFromStream* – load the scheme from the stream.
- *LoadFromFile* - load the scheme from a file.
- *ShowMetrix* - show the statistics on the slice – the number of fields in different regions, the elapsed time for different operations.
- *HaveLayout* - the function returns the true value in case there is at least one field in any regions of the slice.
- *IsMeasureField* - the function checks whether the AIndex field in the ARegion is

a measurement field.

- *SetAllFilter* – select all the values of the AIndex field in the ARegion for display.
- *InverseFilter* - inverse visibility of the field values in the region.
- *SetNoneFilter* - turn off visibility of all the field values in the region.
- *GetFactFields* - get a list of facts built on the Field. If Field = nil, a general list of all facts is created.
- *SetFactFields* - make a list of facts.
- *AddFactFields* - add the facts from the list to the other facts.
- *GetDetailInfoAbs* – receive detailed data on the values at index crossings on the X axis = XAxisIndex, and the Y axis = YAxisIndex, which are set in absolude coordinates.
- *GetDetailInfoVis* - same as above, but in visible coordinates.
- *AddFieldTo* - add dimension based on field AFieldName to the ARegionOfField and set caption ACaption. If a region is a region of facts, the aggregation function AAgrFunc should be specified. AName - name of dimension.
- *AddCalcFieldTo* - add a fact that is calculated by means of AFormula with the AOrder calculations order and the aggregation function AAgrFunc, set caption ACaption. AName - name of fact.
- *InsertFieldTo* - insert dimension based on field AFieldName to the ARegionOfField in the Index position and set caption ACaption. AName - name of dimension.
- *RemoveFieldFrom* - remove dimension based on field AFieldName with caption ACaption from the ARegionOfField. Also You can use dimension name: AName. When the fact is deleted, its aggregation functionAAgrFunc should be specified.
- *MoveBefore* – place the field with AFieldIndex in the ARegionOfField before the field with the AFieldIndexBefore index.
- *AddCountToFact* - add a special “system counter” fact.
- *RemoveCounTfcmFact* - remove the “system counter” fact.
- *FieldsOfRegion - return the list of fields in the ARegionOfField region*.
- *TraverseAxis* - the function of traversal of axis Y if ARowAxis = True or, on the contrary, traversal of axis X from AFirstItem element to ALastItem element. During the traversal the function calls the transferred Proc method with the current cell settings. Also, the methods of getting the column width AGetColWidth and the row height AGetRowHeight may be transferred into the function, in case their values are stored in the structures outside the slice.

The TfcSlice class includes the following properties:

- *DefaultColWidth* - default column width.
- *DefaultRowHeigh*t - default row height.
- *Options* - allow one to set the following slice properties:
  - **mdsoNullIsZero** - the NULL value is taken for 0.
  - **mdsoTotalAsPrevLevel** - the name of the dimension of the previous level is displayed in the final dimension cell. In case this option is not on, the displayed “Total” value will depend on the used localization resources.

- **mdsoAutoFilter** - apply filters immediately. If this option is on, the filtration is performed immediately after the value of a filter for any value is changed. Thus the changes in the general picture caused by the change of this value may be observed. If this option is not on, the changes in the filter should be activated by pressing the Use Filter button or rejected by pressing Cancel.
- **mdsoNoTotalIfOneValue -** Total not visible if only one value
- **mdsoAddTotalPrefix -** Total as 'Total ' plus value from parent level
- **mdsoSaveChartInSchema -** Save Chart Properties With Slice
- **mdsoFieldsMultyUse -** Allow field usage both for Dimensions and Measures
- **mdsoSaveFiltersByValue -** Save Filters By Value
- **mdsoSaveFiltersEnabledValues -** Save in Filters Enabled Values
- **mdsoAllowFilterAllValues** - Allow filter all values in field
- **mdsoCalcOrderByDimOrder** - Use Dimensions order during calculation

- *Cube* - link to the cube.
- *NullStr* - the line that will be inserted instead of the Null variables.
- *UnAssignedStr* - the line that will be inserted instead of the UnAssigned variables.

- *StoredSchema* - stored schema;
- *BuildingData* - slice in calculation process;
- *Active* – displays the slice activity status.
- *ActiveMeasure* – shows whether the measures are active (placed on an axis).
- *SliceFields* - provides access to the fields of the slice.
- *HideRowZeros* - hide zero rows.
- *HideColZeros* - hide zero columns.
- *TypeSortOfXAxis* - type of sorting the X axis.
- *TypeSortOfYAxis* - type of sorting the Y axis.
- *SelectedFact* - selected fact (for sorting by fact).
- *SelectedCol* - selected column (for sorting by fact).
- *SelectedRow* - selected row (for sorting by fact).
- *RegionFieldsCount* - the number of fields in the region.
- *RegionFieldCaption* - caption of the field with AFieldIndex in the ARegion.
- *RegionFieldFilter* - shows whether filtering is performed for the field with AFieldIndex in the ARegion.
- *RegionFieldExpanded* - shows the expansion of the field with AFieldIndex in the ARegion is.
- *XAxisValueAbsIndexInLevel* - absolute coordinate of a visible cell on the X axis with the index AVisIndex on the level ATargetLevel.
- *XAxisValueAbsIndexInLevel* - absolute coordinate of a visible cell on the Y axis with the index AVisIndex on the level ATargetLevel.
- *XAxisValueExpandedAbs* - the AIndex cell expansion on the level ALevel on the X axis in absolute coordinates.
- Y*AxisValueExpandedAbs* - the AIndex cell expansion on the level ALevel on the Y axis in absolute coordinates.
- *XAxisValueExpandedVis* - the AVisIndex cell expansion on the level A Level on the X axis in visible coordinates.

- Y*AxisValueExpandedVis* - the AVisIndex cell expansion on the level A Level on the Y axis in visible coordinates.

- *XAxisValueMeasureValueVis* - the fact index (on the list of facts) for a visible coordinate AVisIndex on the X axis.

- Y*AxisValueMeasureValueVis* - the fact index (on the list of facts)for a visible coordinate AVisIndex on the Y axis.

- *XAxisFullExpanded* - the expansion of ALevel on the X axis.

- Y*AxisFullExpanded* - the expansion of ALevel on the Y axis.

- *XAxisValueVisibleChildCountAbs* - the number of children on the last level for the AIndex cell on the level ALevel on the X axis in absolute coordinates.

- *YAxisValueVisibleChildCountAbs* - the number of children on the last level for the AIndex cell on the level ALevel on the Y axis in absolute coordinates.

- *XAxisAllCountInLevel* - the number of cells on ALevel for the X axis.

- *YAxisAllCountInLevel* - the number of cells on ALevel for the Y axis.

- *XAxisVisibleItemCountInLevel* - the number of visible cells on ALevel for the X axis.

- *YAxisVisibleItemCountInLevel* - the number of visible cells on ALevel for the Y axis.

- *XAxisVisibleItemCount* - the number of the lower-level visible cells on the X axis.

- *YAxisVisibleItemCount* - the number of the lower-level visible cells on the on the Y axis.

- *XAxisFieldOrder* - the sorting order for the Index field on the X axis. True - direct, False - reverse.

- *YAxisFieldOrder* - the sorting order for the Index field on the Y axis. True - direct, False - reverse.

- *XAxisExistsVisibleInLevel* - checking for the existence of at least one value on the ALevel of the X axis.

- *YAxisExistsVisibleInLevel* - checking for the existence of at least one value on the ALevel of the Y axis.

- *XAxisValueTypeAbs* - the type of AIndex cell on the level ALevel on the X axis in absolute coordinates.

- *YAxisValueTypeAbs* - the type of AIndex cell on the level ALevel on the Y axis in absolute coordinates.

- *XAxisValueTypeVis* - the type of AVisIndex cell on the level ALevel on the X axis in absolute coordinates.

- *YAxisValueTypeAbs* - the type of AVisIndex cell on the level ALevel on the Y axis in absolute coordinates.

- *XAxisValueParentIndexVis* - the absolute coordinate of the visible cell AVisIndex on ALevel of the X axis.

- *YAxisValueParentIndexVis* - the absolute coordinate of the visible cell AVisIndex on ALevel of the Y axis.

- *XAxisIncludeStateAnyAbs* - checking for the presence of a transferred status of any kind for the cell on the X axis in absolute coordinates.

- *YAxisIncludeStateAnyAbs* - checking for the presence of a transferred status of any kind for the cell on the Y axis in absolute coordinates.

- *XAxisIncludeStateAnyVis* - checking for the presence of a transferred status of any kind for the cell on the X axis in visual coordinates.

- *YAxisIncludeStateAnyVis* - checking for the presence of a transferred status of any kind for the cell on the Y axis in visual coordinates.

- *XAxisValueAbs* - the value of the AIndex cell on the level ALevel on the X axis in absolute coordinates.

- *YAxisValueTypeAbs* - the value of the AIndex cell on the level ALevel on the Y axis in absolute coordinates.

- *XAxisValueVis* - the value of the AIndex cell on the level ALevel on the X axis in visible coordinates.

- *YAxisValueVis* - the value of the AIndex cell on the level ALevel on the Y axis in visible coordinates.

- *XAxisColWidth* - the column width ACol on the last level on the X axis in visible coordinates.

- *YAxisColWidth* - the column width ACol on the Y axis.

- *XAxisRowHeight* - the row height ARow on the last level on the X axis in visible coordinates.

- *YAxisRowHeight* - the row height ARow on the Y axis.

- *MeasureIndex* - the fact index for the data cell with visible ACol and ARow coordinates.

- *DataVariantAbs* - the value of the cell in the form of a Variant in absolute coordinates.

- *DataVariantVis* - the value of the cell in the form of a Variant in visible coordinates.

- *DataStringAbs* - the value of the cell in the form of a formatted line in absolute coordinates.

- *DataStringVis* - the value of the cell in the form of a formatted line in visible coordinates.

- *DataPercentStringVis* - the percent value of the cell in the form of a formatted line in visible coordinates.

- *DataPercentVariantVis* - the percent value of the cell in the form of a Variant in visible coordinates.

- *DataAlignmentAbs* - fact values alignment by the absolute index AIndex.

- *DataAlignmentVis* - fact values alignment by the visible index AVisIndex.

- *CapXFields* - the array of fields on the X axis.

- *CapYFields* - the array of fields on the Y axis.

- *CapPageFields* - the array of fields in the filters region.

- *CapFactsFieldsAbs* - the array of fact-fields.

- *CapFactsFieldsVis* - the array of visible fact-fields.

- *CapFactsFieldIndexVis* – the absolute index of a visible fact-field.

- *VisibleCapFactsFieldsCount* - the number of visible fact-fields.

- *XAxisLevelCount* - the number of levels on the X axis.

- *YAxisLevelCount* - the number of levels on the Y axis.

- *XAxisVisibleLevelCount* - the number of visible levels on the X axis.

- *YAxisVisibleLevelCount* - the number of visible levels on the Y axis.

- *UniqueCount* - the number of unique values for a AIndex field in the ARegion.

- *UniqueFilter* - checking whether the unique value AUniqueIndex for the AIndex field in the ARegion is visible.
- *UniqueCaption* - the caption of the unique value AUniqueIndex for the AIndex field in the ARegion.
- *SliceUniques* - direct access to the unique values of the slice.
- *FieldCount* - the number of all fields in the slice.
- *MeasureIndexInY* - the index of the Measurement field on the Y axis.
- *MeasureIndexInX* - the index of the Measurement field on the X axis.
- *MeasureField* - the link to the Measurement field.
- *Locked* - informs on the ongoing calculations inside the slice.
- *XAxis* - direct access to the X axis.
- *YAxis* - direct access to the Y axis.
- *FilterRecCount* - count of filtered records.
- *Filters* - Filter Manager.
- *SourceRec* - record of sources data.
- *PopUpFieldListWidth* - width of popup fields list.
- *PopUpWidthDefault* - default width of popup list.
- *FieldsOrder* - order type in fields list.
- *ShowSplitFieldsInFieldList* - show split fields in fields list.
- *StoreInDFM* - store schema DFM.

The following events are defined in the TfcSlice class:

- *OnFieldsListSortCompare* - compare field names for sort fields list;
- *OnInterpreterCreated* - allow add users functions in Interpreter;
- *OnStartChange* - start change in Slice;
- *OnStopChange* - stop change in Slice;
- *OnSliceChanged* - slice changed;
- *OnBeforeRemoveSliceFieldFromRegion* - Allow to block changes;
- *OnBeforeAddedSliceFieldToRegion* - Allow to block changes;
- *OnAfterAddedSliceFieldToRegion* - Allow to set properties of added field;

Chapter V

The Grid

TfcGrid (the grid) is a visual interactive component representing multidimensional data in the form of a 2D table with two hierarchical headers (horizontal and vertical). The grid works in connection with TfcSlice (the slice), providing the opportunity for displaying and modifying the slice's structure.

```
TfcGrid = class(TfcCustomGrid)
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    function ExecuteAction(Action : TBasicAction) : Boolean; override;
    procedure AdjustClientRect(var ARect: TRect); override;
    procedure UpdateGrid; override;
    procedure DragDrop(Source: TObject; X, Y: Integer); override;
    procedure DragCanceled; override;
    procedure Open;
    procedure Close;
    // save/load dialogs
    procedure SaveSchemeToFile;
    procedure LoadSchemeFromFile;
    procedure SaveCubeToFile;
    procedure LoadCubeFromFile;
    procedure DoExportTo(const AFormat: TfcExportFormat; ACloneMasterValue:
boolean = False);
    procedure UpdateSelection;
    procedure SelectCell(ACol, ARow: integer; AMakeVisible: boolean;
ACalcRowHeight: Boolean = False);
    property FieldEditor: TfcFieldEditor read FFieldEditor;
    property SelectedFact: Integer read GetSelectedFact;
    property CanOperate: Boolean read GetCanOperate;
    property GridCanvas: TCanvas read GetCanvas;
  published
    property OnChange: TNotifyEvent read FOnChange write FOnChange;
    property OnGetDimDetail: TfcGridGetDimDetailEvent read FOnGetDimDetail write
FOnGetDimDetail;
    property OnDataDblClick: TfcGridDataDblClickEvent read FOnDataDblClick write
FOnDataDblClick;

    property OnGetFieldImageIndex: TfcGridGetFieldImageIndexEvent read
FOnGetFieldImageIndex write FOnGetFieldImageIndex;
    property OnGetAxisImageIndex: TfcGetAxisImageIndexEvent read
FOnGetAxisImageIndex write FOnGetAxisImageIndex;
    property OnGetCellImageIndex: TfcGetCellImageIndexEvent read
FOnGetCellImageIndex write FOnGetCellImageIndex;
```

```
    property OnUpdateSelection: TNotifyEvent read FOnUpdateSelection write
FOnUpdateSelection;
    property OnDrawCell: TfcGridDrawCellEvent read FOnDrawCell write
FOnDrawCell;
    property OnDrawAxisItem: TfcGridDrawAxisItemEvent read FOnDrawAxisItem
write FOnDrawAxisItem;
    property OnSaveCube: TNotifyEvent read FOnSaveCube write FOnSaveCube;
    property OnSaveScheme: TNotifyEvent read FOnSaveScheme write
FOnSaveScheme;
    property OnLoadCube: TNotifyEvent read FOnLoadCube write FOnLoadCube;
    property OnLoadScheme: TNotifyEvent read FOnLoadScheme write
FOnLoadScheme;
    property OnExportTo: TfcExportEvent read FOnExportTo write FOnExportTo;

    property DefaultSchemePath: string read FDefaultSchemePath write
FDefaultSchemePath;
    property DefaultCubePath: string read FDefaultCubePath write FDefaultCubePath;
    property DefaultExportPath: string read FDefaultExportPath write
FDefaultExportPath;

    property DefaultRowHeight: Integer read GetDefaultRowHeight write
SetDefaultRowHeight default 18;
    property DefaultColWidth: Integer read GetDefaultColWidth write
SetDefaultColWidth default 80;
    property DefaultItemWidth: Integer read FDefaultItemWidth write
SetDefaultItemWidth default 100;
    property Options: TfcGridOptions read FOptions write SetOptions default
GridDefaultOptions;

    // image lists
    // FieldsImages - images of field captions
    // AxisImages - images of axis values
    // CellImages - images of cell values
    property FieldsImages: TCustomImageList read FFieldsImages write
SetFieldsImages;
    property AxisImages: TCustomImageList read FAxisImages write SetAxisImages;
    property CellImages: TCustomImageList read FCellImages write SetCellImages;
    property Active;
    property Styles;
    property Slice;
    property Align;
    property Anchors;
    property Constraints;
    property Ctl3D;
    property Enabled;
```

```
  property Font;
  property ParentCtl3D;
  property ParentFont;
  property PopupMenu;
  property TabOrder;
  property TabStop;
  property Visible;
end;
```

The TfcGrid class includes the following methods:

- *Open* - open the data source (i.e., the cube) linked with the grid.
- *Close* - close the data source linked with the grid.
- *SaveSchemeToFile* - save the data layout scheme to file though the dialog.
- *LoadSchemeFromFile* - load the data layout scheme from file though the dialog.
- *SaveCubeToFile* - save the data and the layout scheme to file though the dialog.
- *LoadCubeFromFile* - load the data and the layout scheme from file though the dialog.
- *DoExportTo* - export the data to file through the dialog[1].
- *UpdateSelection* - update the value of the SelectedCol, SelectedRow, and SelectedFact properties in the TfcSlice linked with the grid.
- *SelectCell* - select cell in grid.

Endnote 1' Structurally the export into .xls and .doc is performed though html documents.

The TfcGrid class includes the following properties:

- *FieldsImages* - the ImageList used for drawing the fields.
- *AxisImage* - the ImageList used for drawing the axes values.
- *CellImages* - the ImageList used for drawing the values in the data region.
- *SelectedFact* - the fact index of the current active cell.
- *DefaultRowHeigh*t - default row height.
- *DefaultColWidth* - default column width.
- *DefaultItemWidth* - default header width with the mdgoYCaptionHooked turned off, otherwise the value of DefaultColWidth is used.
- *CanOperate* - information on the possibility of performing any operations on the grid.
- *GridCanvas* - Grid canvas.
- *Options* include the following properties of the grid:
  - **mdgoYCaptionHooked** - caption (field) width of the Y axis, the same with the Y axis columns.
  - **mdgoResizeOnChange** - change the size of the region of the Y axis when the set of fields changes in the region.
  - **mdgoServiceRegion** - show the service region.
  - **mdgoToolRegion** - show the region with the choice of fields.

- **mdgoShowHints** - show hints when mouse over axes cells.
- **mdgoRightClickSelect** - highlight cells with right-click of the mouse.
- **mdgoShowCaption** - show the cube caption.
- **mdgoDrillThroughOnDblClick** - show a detailed report down to original data with a double-click on the cell.
- **mdgoYAxisScroller** - show the scroller for the Y axis fields (good with a large number of fields).
- **mdgoAutoHeight** - automatically set the height for cell values on the axes.
- **mdgoDisableXAxisChange** - disable changes of the X axis.
- **mdgoDisableYAxisChange** - disable changes of the Y axis. //
- **mdgoDisablePageChange** - disable changes in the filter area. /
- **mdgoDisableDataChange** - disable changes in the data area.
- **mdgoCloneMasterValueOnExport -** Clone Master Value On Export
- **mdgoChangeOrderByClick** - Change Order By One Click

- Active - if set as True starts the Open method, when False starts Close method. Shows the activity status of the grid.
- *Slice* - the link to the slice.
- *Styles* - the styles for displaying various regions of the grid.
- *DefaultSchemePath* - default path to save schema.
- *DefaultCubePath* - default path to save cube.
- *DefaultExportPath* - default path to save exported files.

The TfcGrid class includes the following events:

- *OnChange* - is called when the grid structure is altered.
- *OnGetDimDetail* - is called when the **mdgoShowHints** option is turned on in order to get additional information on the dimension value.
- *OnDataDblClick* - is called with a double click on a data cell.
- *OnGetFieldImageIndex* - receives the image index from *FieldsImages* to be placed on the margins.
- *OnGetAxisImageIndex* - receives the image index from *AxisImages* to be placed on the axes cells.
- *OnGetCellImageIndex* - receives the image index from *CellImages* to be placed on the data cells.
- *OnUpdateSelection* - selection changed.
- *OnDrawCell* - draw data cell.
- *OnDrawAxisItem* - draw axis cell.
- *OnSaveCube* - save cube.
- *OnSaveScheme* - save schema.
- *OnLoadCube* - load cube.
- *OnLoadScheme* - load schema.
- *OnExportTo* - export.

Chapter VI

The Chart

TfcChart is a visual component for representing multidimensional data in the form of graphs and charts. The TfcChart component is based on a chart from the TeeChart package and, consequently, it inherits all the corresponding properties and methods.

```
TfcCustomChart = class(TCustomChart)
public
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
  function ExecuteAction(Action: TBasicAction): Boolean; override;
  procedure Updated; override;
  procedure FillFacts(Items: TStrings);
  procedure AddToUpdates(Obj: TObject);
  procedure RemoveFromUpdates(Obj: TObject);
  procedure SaveToStream(Stream: TStream);
  procedure LoadFromStream(Stream: TStream);
  procedure SaveTemplate(DefaultDirectory: string = '');
  procedure LoadTemplate(DefaultDirectory: string = '');

  property Slice: TfcSlice read FSlice write SetSlice;
  property Active: Boolean read FActive write SetActive;
  property Is1D: Boolean read FIs1D;

  property SeriesType: TChartSeriesClass read FSeriesType write SetSeriesType;
  property AxisType: TfcChartAxisType read FAxisType write SetAxisType;
  property LogType: TfcChartLogType read FLogType write SetLogType;
  property MarksShowStyle: TfcMarksShowStyle read FMarksShowStyle write
    SetMarksShowStyle;
  property AllSeriesMarksStyle: TSeriesMarksStyle read FAllSeriesMarksStyle
    write SetAllSeriesMarksStyle;
  property CanSelectFact: Boolean read FCanSelectFact;
  property SelectedFact: Integer read FSelectedFact write SetSelectedFact;
  // events
  property OnChangeChart: TNotifyEvent read FChangeChart write FChangeChart;
end;
```

The TfcChart class includes the following methods:

- *FillFacts* - fill out the list with fact names, which may be automatically displayed in the chart.
- *AddToUpdates* - add an object to the list of listeners of updates.
- *RemoveFromUpdates* - remove an object from the list of listeners of updates.
- *SaveToStream* - save the chart settings to stream.
- *LoadFromStream* - load up the chart settings from stream.

- *SaveTemplate* - save the chart settings to file by means through the dialog.
- *LoadTemplate* - load the chart settings from file thought the dialog.

The TfcChart class includes the following properties:

- *Slice* - the link to the slice.
- *Active* - shows if the chart is active, including the information on whether the chart structures are filled out with data.
- *Is1D* - informs on the number of chart's dimensions (one/two).
- *SeriesType* - the type of chart series.
- *AxisType* specifies what will go in the series and what will be put in the legend.«

With atByCols in a series it brings out values from the X axis, otherwise – the Y axis.

- *MarksShowStyle* - hints displaying style .
  - **ssNone** - no hints;
  - **ssHint** - hints popping up at mouse over the series.
  - **ssAlways** - the hints that are always visible.
- *CanSelectFact* - informs on the possibility of setting the fact index which is to be displayed in the chart.
- *SelectedFact* - the fact index displayed in the chart.

The TfcChart class includes the following events:

- *OnChangeChart* is called when the chart structure is altered.

Chapter VII

Toolbars

In order to manage the status of the grid, a number of corresponding toolbars was included in the components set. They are set by just specifying the component which they would manage. For the grid toolbar (TfcToolBar) it is necessary to fill out the Grid field, and for the chart instruments (TfcChartToolBar) – the chart field.

```
TfcToolBar = class(TToolBar)
 published
  property Grid : TfcGrid read FGrid write SetGrid;
  property Options : TfcToolBarButtons read FButtons write SetButtons;
 end;

 TfcChartToolBar = class(TToolBar)
 published
  property Chart: TfcCustomChart read FChart write SetChart;
        end;
```

Chapter VIII

The Grid Report

TfcGridReport is a visual component for exporting the grid into FastReport.

```
TfcGridReport = class(TComponent)
 public
  procedure PrintPreview;
  function Print: Boolean;

  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
 published
  property Grid: TfcGrid read FGrid write FGrid;
  property PaintSizes: TfrcCubeCrossPaintSizes read FPaintSizes write SetPaintSizes;
 end;
```

The TfcGridReport class includes the following methods:
- *Print* - printing on the printer.

The TfcGridReport class includes the following properties:

- *Grid* - link to the grid
- *PaintSizes* - setting the size of the cells while building the report (see the Grid Report chapter.)

Chapter

Register

TfrcComponents is an empty component necessary to automatically add the frxFCComponents.pas on the list of uses. Use this component for securing an opportunity for FastCube (TfrcCube and TfrcCrossView) in the FastReport report.

```
TfrcComponents = class(TComponent)
 public
   constructor Create(AOwner: TComponent); override;
   destructor Destroy; override;
   function GetDescription: String;
         end;
```

In order to include the FastCube objects in the в FastReport, place the TfrcComponent object on the form.

Chapter

Cube View

TfrcCube = class(TfrxDialogComponent)

The **Cube View** is the main data storage for creating reports, which is used in FastReport as the data source for grids.

TfrcCube is the holder of the TfcCube object and can use its functions.

```
TfrcCube = class(TfrxDialogComponent)
 public
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
  class function GetDescription: String; override;
  property Cube: TfcCube read GetCube write SetCube;
 published
  property Dataset: TfrxDataset read FDataset write SetDataset;
  property FileName: String read GetFileName write SetFileName;
  property Active: Boolean read GetActive write SetActive;
    end;
```

In order to build reports with a grid one needs to put a TfrcCube object on the data sheet from the FastCube section. Then one should either connect a data set to this object or type in the name of a previously created cube file through the object inspector .

In order to build a grid, the cube view should be active. To do it, set the Active property as True.

One cube view can serve as the data source for an unlimited number of grids with different structures at the same time.

The *TfrcCube* class includes the following properties:

DataSet: *TfrxDataset* – a FastReport data set. At the moment only TfrxDBDataset are supported.

FileName: String - the name of a saved cube.

Active: Boolean - shows whether the cube is active or not. If *Active* = True, then multidimensional data structures are created inside the cube. These structures are filled up either with the values from the Dataset, or with the data from the cube file. If Active = False, the cube becomes inactive, and its internal structures are not filled up with anything.

Cube: TfcCube - provides access to the cube object.

Chapter XI

Cross View

TfrcCrossView - (cross view) is used for setting the slice properties and the report layout.

```
TfrcCrossView = class(TfrxView)
 public
   procedure ImportColorFromGrid(AfcGrid : TfcGrid);
   procedure ClearColorFromGrid;
   property Slice: TfcSlice read FSlice write SetSlice;
   property DotMatrix: Boolean read FDotMatrix;
   property OnBeforePrintCell: TfrxOnPrintCellEvent read FOnBeforePrintCell write
FOnBeforePrintCell;
   property OnBeforePrintColumnHeader: TfrxOnPrintHeaderEvent read
FOnBeforePrintColumnHeader write FOnBeforePrintColumnHeader;
   property OnBeforePrintRowHeader: TfrxOnPrintHeaderEvent read
FOnBeforePrintRowHeader write FOnBeforePrintRowHeader;
   property CellMemos[Index: Integer]: TfrxCustomMemoView read GetCellMemos;
   property CellHeaderMemos[Index: Integer]: TfrxCustomMemoView read
GetCellHeaderMemos;
   property ColumnMemos[Index: Integer]: TfrxCustomMemoView read
GetColumnMemos;
   property ColumnTotalMemos[Index: Integer]: TfrxCustomMemoView read
GetColumnTotalMemos;
   property CornerMemos[Index: Integer]: TfrxCustomMemoView read
GetCornerMemos;
   property RowMemos[Index: Integer]: TfrxCustomMemoView read GetRowMemos;
   property RowTotalMemos[Index: Integer]: TfrxCustomMemoView read
GetRowTotalMemos;
   property RowLevels: Integer read GetRowLevels;
   property ColumnLevels: Integer read GetColumnLevels;
   property CellLevels: Integer read GetCellLevels;
   property DefaultColWidthInternal: integer read GetDefaultColWidthInternal;
   property DefaultRowHeightInternal: integer read GetDefaultRowHeightInternal;
   property ColWidth[ACol: Integer]: integer read GetColWidth write SetColWidth;
   property RowHeight[ARow: Integer]: integer read GetRowHeight write
SetRowHeight;
 published
   property GapX: Integer read FGapX write FGapX default 3;
   property GapY: Integer read FGapY write FGapY default 3;
   property Border: Boolean read FBorder write FBorder default True;
   property Cube: TfrcCube read FCube write SetCube;
   property DownThenAcross: Boolean read FDownThenAcross write
FDownThenAcross;
```

```
  property RepeatHeaders: Boolean read FRepeatHeaders write FRepeatHeaders
default True;
  property ShowColumnHeader: Boolean read FShowColumnHeader write
SetShowColumnHeader default True;
  property ShowRowHeader: Boolean read FShowRowHeader write
SetShowRowHeader default True;
  property ShowNames: Boolean read FShowNames write SetShowNames default
True;
  property OnPrintCell: TfrxPrintCellEvent read FOnPrintCell write FOnPrintCell;
  property OnPrintColumnHeader: TfrxPrintHeaderEvent read
FOnPrintColumnHeader write FOnPrintColumnHeader;
  property OnPrintRowHeader: TfrxPrintHeaderEvent read FOnPrintRowHeader write
FOnPrintRowHeader;
  property NextCross: TfrcCrossView read FNextCross write FNextCross;
  property NextCrossGap: Extended read FNextCrossGap write FNextCrossGap;
  property KeepTogether: Boolean read FKeepTogether write FKeepTogether default
False;
  property PaintSizes: TfrcCubeCrossPaintSizes read FPaintSizes;
    end;
```

TfrcCrossView is a descendant of TfrxView and inherits its properties. Several important properties are additionally defined in this class:

- *GapX: Integer* - in the grid cells, the text indents from the left and right boundaries of a cell.

- *GapY: Integer* - in the grid cells, the text indents from the top and bottom boundaries of a cell.

- *Border: Boolean* - the property defines whether a frame should be drawn around the grid cells. This property is True by default.

- *Cube:*

TfrcCube - Cube view is the main data storage for the grid.

- *DownThenAcross: Boolean* - The property defines the way of splitting a big grid into several pages. If it is set as False (by default), the grid is split first horizontally, then vertically.

- *RepeatHeaders: Boolean* - The property defines whether there is a need to repeat the grid headers on every page of the report. True by default.

- *ShowColumnHeader: Boolean* - the property defines whether the grid column headers should be printed. This property is True by default.

- *ShowRowHeader: Boolean* - the property defines whether the grid row headers should be printed. This property is True by default.

- *ShowNames: Boolean* - the property defines whether the grid cell headers should be printed. This property is True by default.

- *OnPrintCell: TfrxPrintCellEvent* -

- *OnPrintColumnHeader: TfrxPrintHeaderEvent* -

- *OnPrintRowHeader: TfrxPrintHeaderEvent* -

- *NextCross: TfrcCrossView* - pointer to the next cross object that will be displayed next to this one.

- *NextCrossGap: Extended* - the gap between two crosstabs located next to each other (see NextCross).

- *KeepTogether: Boolean* - the property that prohibits to split the cells vertically when the page is split. If the grid does not fit on the free space on the page, it is transferred to the next page. False by default.

- *PaintSizes: TfrcCubeCrossPaintSizes* - setting the size of the cells while building the report .

```
TfrcCubeCrossPaintSizes = class(TPersistent)
  published
    property AutoSizeStyle: TfrcCubeCrossAutoSizeStyle read
  FAutoSizeStyle write SetAutoSizeStyle;
    property MaxColWidth: Integer read FMaxColWidth write
  FMaxColWidth;
    property DefaultRowHeight: integer read FDefaultRowHeight write
  SetDefaultRowHeight;
    property DefaultColWidth: integer read FDefaultColWidth write
  SetDefaultColWidth;
end;
```

- *AutoSizeStyle: TfrcCubeCrossAutoSizeStyle* - the style for automatic settings of the grid print size.

May have the following values:

- ssDefault - uses DefaultColWidth, DefaultRowHeight
- ssBySlice - by the sizes of the "live" table in the editor
- ssAutoColWidth - automatic width calculation
- ssAutoColWidthRestrict - automatic width calculation limited by the value
- ssAutoRowHeigh - automatic height calculation
- ssByMemoSize – by the size of the memo-objects built in the object

- *MaxColWidth: Integer* - width limitation when AutoSizeStyle = ssAutoColWidthRestrict

- *DefaultRowHeight: - integer* - default row height for drawing the grid

- *DefaultColWidth: - integer* - default column width for drawing the grid

Public properties:

- *Slice: TfcSlice* - the slice used in building the report.

- *DotMatrix: Boolean* - the property defines whether or not the report is a matrix one.

The following group of properties gives access to the template cells of the corresponding areas of the grid

*CellMemos[**Index**: Integer]: TfrxCustomMemoView*
*CellHeaderMemos[**Index**: Integer]: TfrxCustomMemoView*
*ColumnMemos[**Index**: Integer]: TfrxCustomMemoView*
*ColumnTotalMemos[**Index**: Integer]: TfrxCustomMemoView*

*CornerMemos[**Index**: Integer]: TfrxCustomMemoView*
*RowMemos[**Index**: Integer]: TfrxCustomMemoView*
*RowTotalMemos[**Index**: Integer]: TfrxCustomMemoView*

- *RowLevels: Integer* - the number of levels on the row axis.
- *ColumnLevels: Integer* - the number of levels on the column axis.
- *CellLevels: Integer* – the number of facts.
- *DefaultColWidthInternal: - integer* - default column width depending on the value of PaintSizes.AutoSizeStyle.
- *DefaultRowHeightInternal: - integer* - default row height depending on the value of PaintSizes.AutoSizeStyle.

ColWidth[ACol: Integer]: Integer – previously calculated column width.
RowHeight[ARow: Integer]: Integer – previously calculated row height.

Public properties:

- **procedure** ImportColorFromGrid(AfcGrid : TfcGrid) - allows to import the colour scheme from the grid.
- **procedure** ClearColorFromGrid - clears the colour scheme imported from the grid.