My Visual Database

Table of contents

About My Visual Database

A simple development environment allows you to create databases without the help of specialists or the need for programming skills. MVD will enable you to create a self-contained database application that operates on Windows XP, 7, 8 and 10. Databases can be simple telephone directories or basic accounting systems.

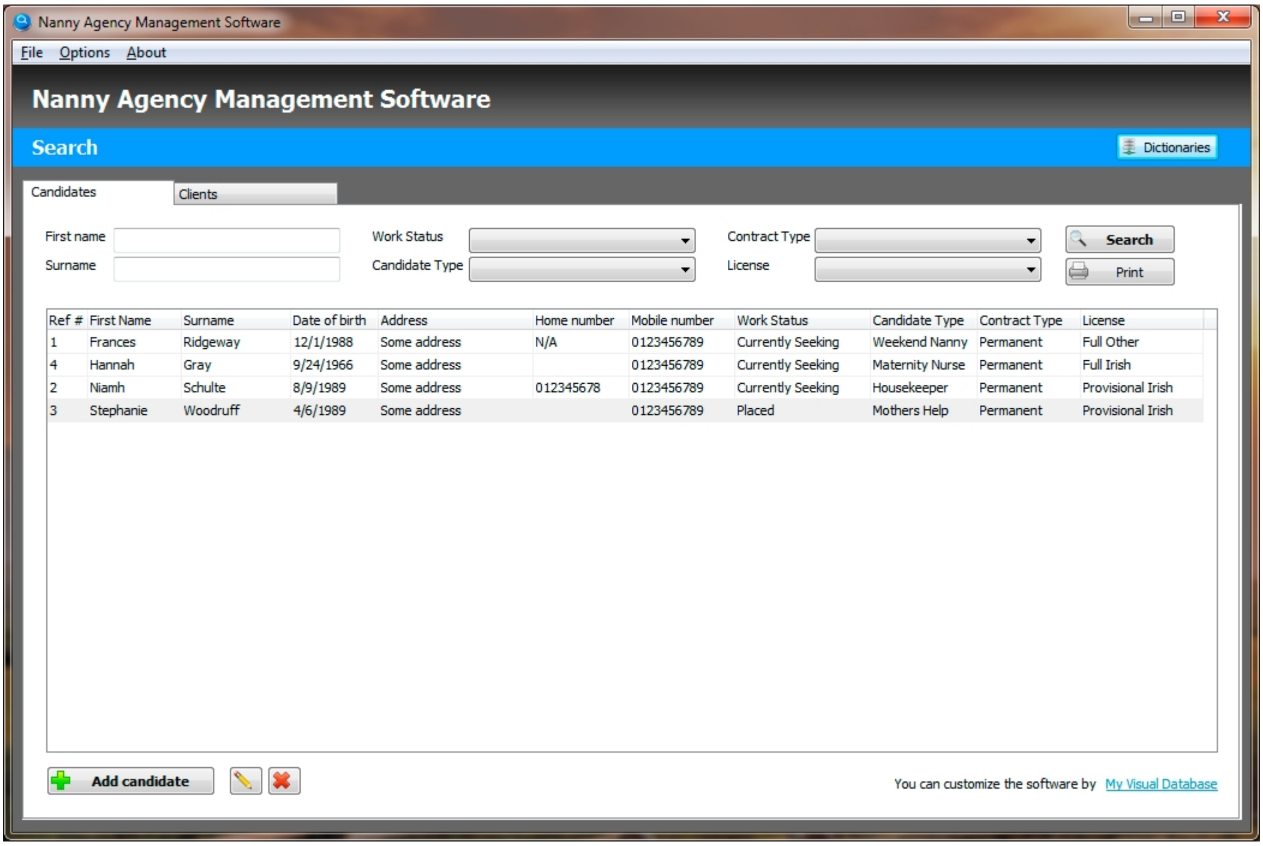
The result of your work will be a full-fledged Windows application that does not require installation and third-party components and can work even with a flash drive.

By default, the application you create uses a fairly simple but at the same time reliable SQLite DBMS.
As a rule, SQLite is intended for single-user work, but multiuser work in the local network through a shared folder is also allowed.

If necessary, your application can use a MySQL database. This database is multi-user and perfectly suited for working over the Internet.

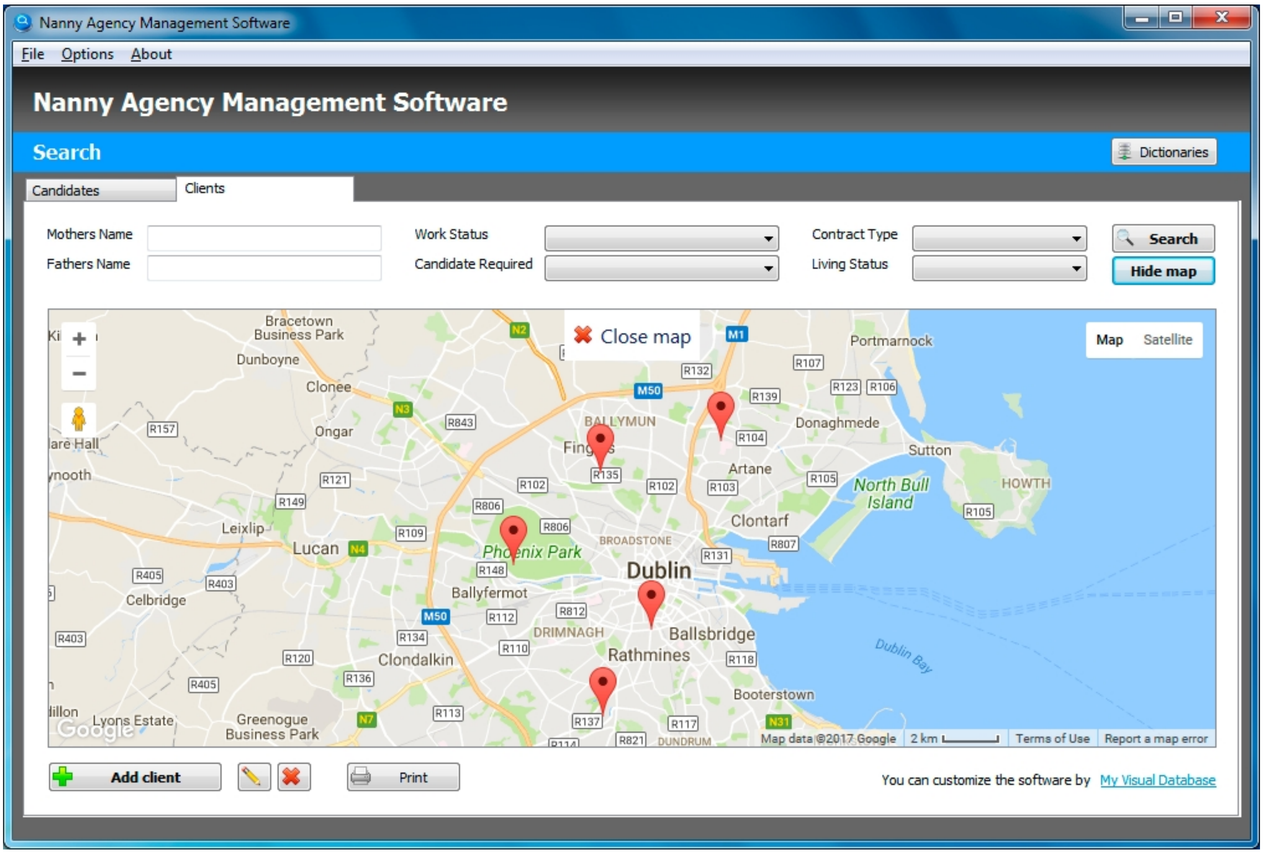
For advanced users and programmers there are scripts (Object Pascal), with many built-in functions and classes that will allow you to implement any functionality of your future application.

The figure below shows an example of what a ready-made application created in My Visual Database might look like



Ref #	First Name	Surname	Date of birth	Address	Home number	Mobile number	Work Status	Candidate Type	Contract Type	License
1	Frances	Ridgeway	12/1/1988	Some address	N/A	0123456789	Currently Seeking	Weekend Nanny	Permanent	Full Other
4	Hannah	Gray	9/24/1966	Some address		0123456789	Currently Seeking	Maternity Nurse	Permanent	Full Irish
2	Niamh	Schulte	8/9/1989	Some address	012345678	0123456789	Currently Seeking	Housekeeper	Permanent	Provisional Irish
3	Stephanie	Woodruff	4/6/1989	Some address		0123456789	Placed	Mothers Help	Permanent	Provisional Irish

You can customize the software by My Visual Database



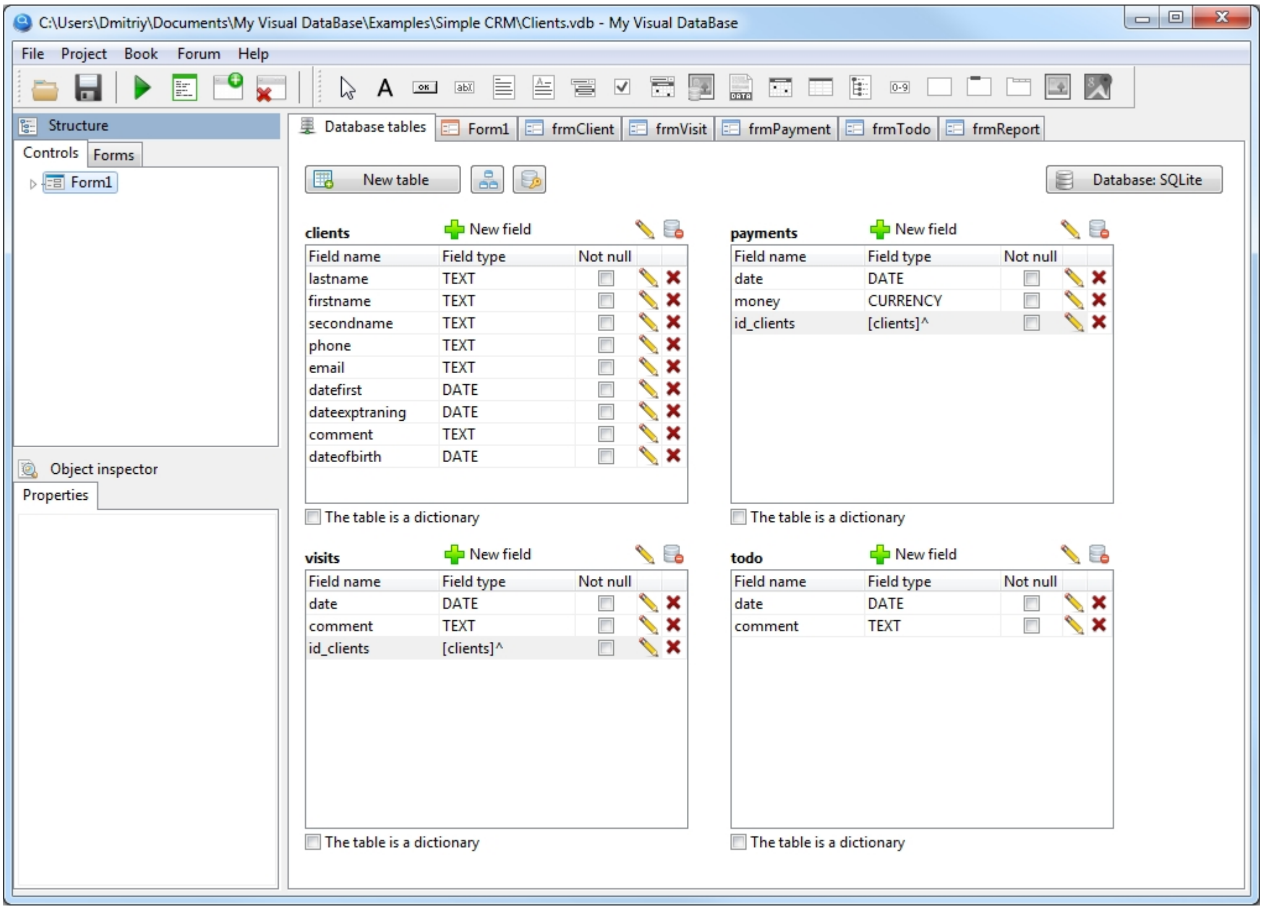
Other application examples can be found here: http://myvisualdatabase.com/products.html

How it works?

Here you will learn the basic principles that are used when creating applications using My Visual Database

1. Creating a database structure.

Since your application will be designed to work with the database, you need to create a structure of your database, which will store the information. Examples of this information may be: Clients, Payments, Visits, etc.



clients

Field name	Field type	Not null
lastname	TEXT	□
firstname	TEXT	□
secondname	TEXT	□
phone	TEXT	□
email	TEXT	□
datefirst	DATE	□
dateexptraning	DATE	□
comment	TEXT	□
dateofbirth	DATE	□

□ The table is a dictionary

payments

Field name	Field type	Not null
date	DATE	□
money	CURRENCY	□
id_clients	[clients]^	□

□ The table is a dictionary

visits

Field name	Field type	Not null
date	DATE	□
comment	TEXT	□
id_clients	[clients]^	□

□ The table is a dictionary

todo

Field name	Field type	Not null
date	DATE	□
comment	TEXT	□

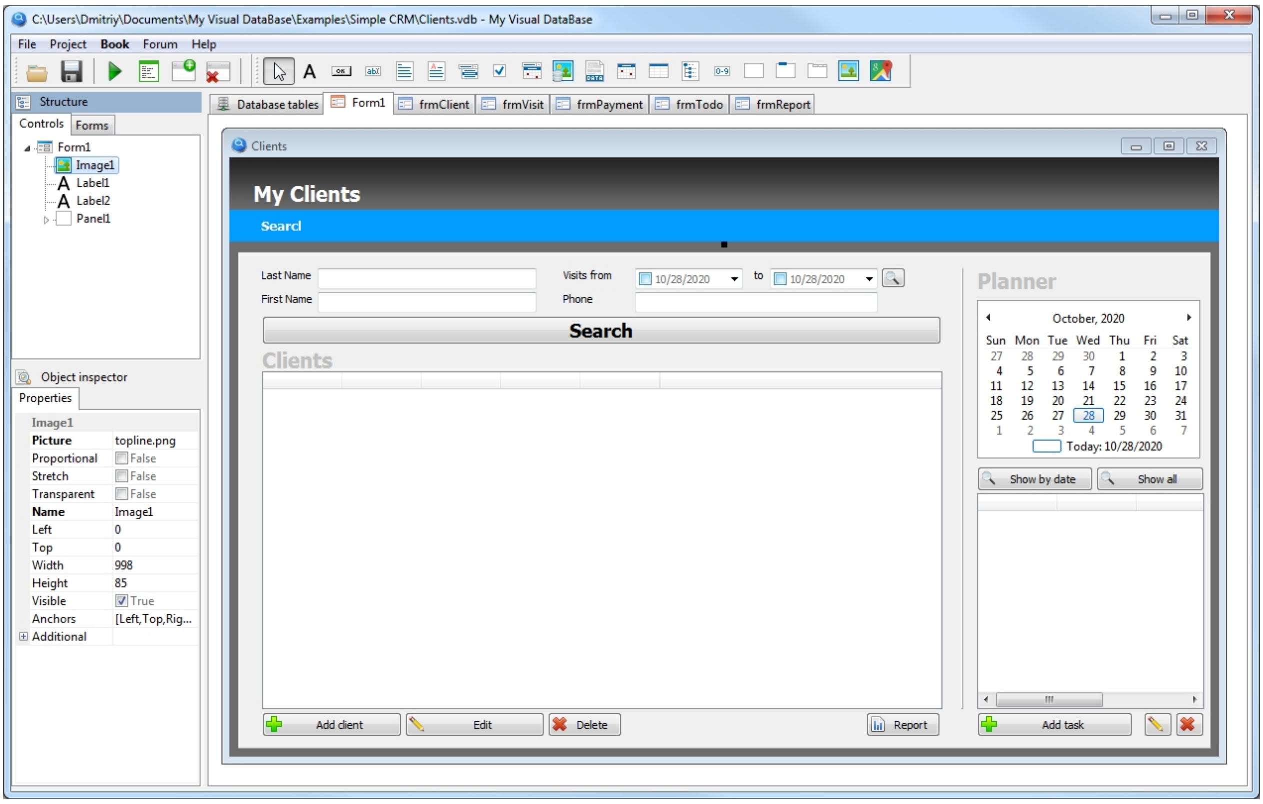
□ The table is a dictionary

To correctly create a database structure, you need to know the basic principles of database design. If you have no experience in creating a database, I strongly recommend that you read the material on this link, namely, the first chapter "1.Introduction".

For more information on how to create a database structure in My Visual Database, see "Database design".

2. Creating a user interface

In order to manage the data, you need to create a user interface. Using the user interface, you can: search, print, calculate, create, edit or delete information from the database. The user interface is a set of forms and components.



C:\Users\Dmitriy\Documents\My Visual DataBase\Examples\Simple CRM\Clients.vdb - My Visual DataBase

Image1	
Picture	topline.png
Proportional	□ False
Stretch	□ False
Transparent	□ False
Name	Image1
Left	0
Top	0
Width	998
Height	85
Visible	☑ True
Anchors	[Left,Top,Rig...
⊞ Additional	

The first form is the main one (Form1), which is what you will see when you launch your application.

Usually, the first (main) form has components that allow you to find information and display it in a table. Also on the form there are buttons to create a new record in the database or to edit/delete an existing one. Creation or editing of records is performed on other forms. This user interface principle is recommended, but nothing prevents you from using a different approach.

Next, you need to configure the components on the form. For buttons, you need to select an action, for example: Search, Create record, Show record, Delete record, etc.
As a rule, it is enough to specify a table name and a field in the database for the components intended to enter information, thus the component communicates with the database.

If the built-in actions for buttons are not enough for you, using scripts, you can implement almost any functionality in the Object Pascal programming language, read more about it in the Scripts section.

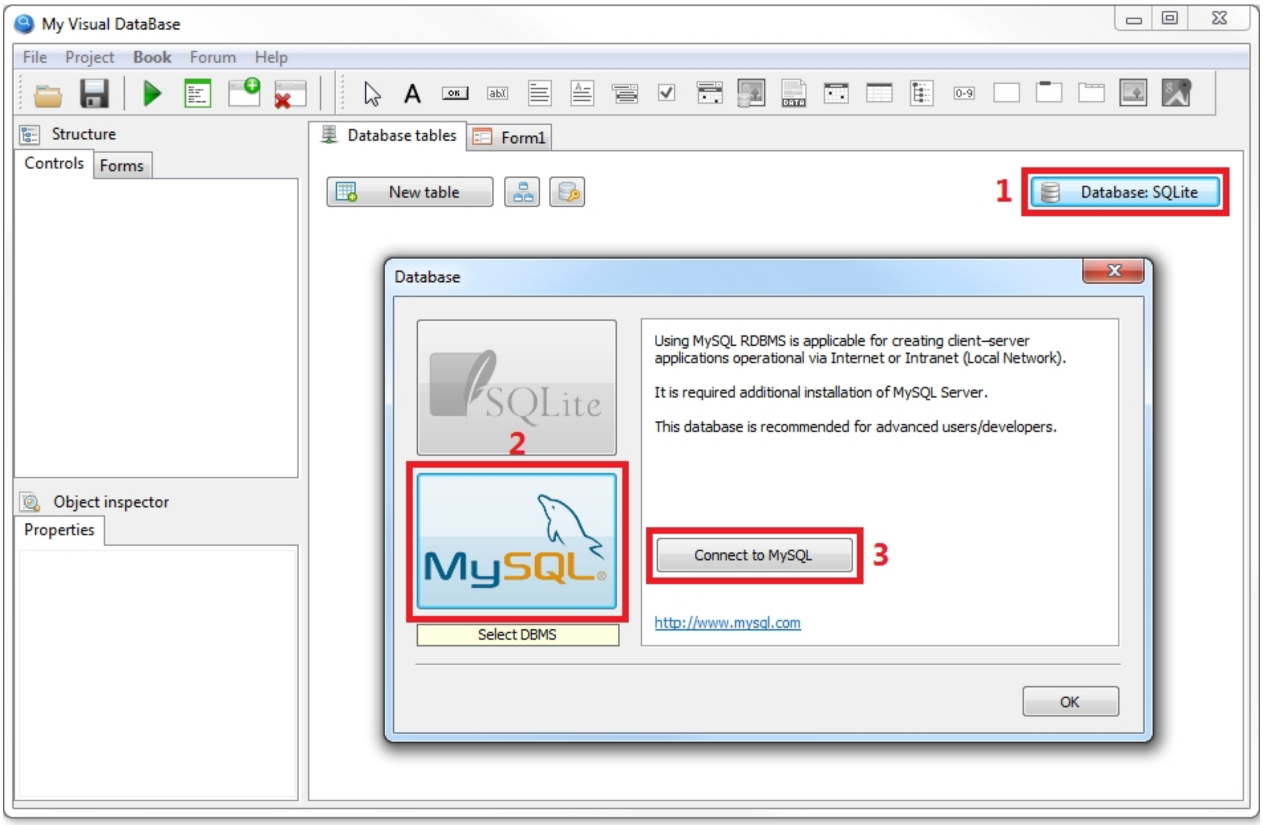
Using DBMS MySQL

By default, the application you create uses a fairly simple but at the same time reliable SQLite database. As a rule, SQLite is intended for single-user work, but multi-user work in the local network through a shared folder is also acceptable.

If necessary, your application can use a MySQL database. This database is multi-user and perfectly suited to work over the Internet, but its use requires that you have the basic skills to configure it.

DBMS MySQL is a free product that you can either install on your computer (https://dev.mysql.com/downloads/mysql/), or purchase web hosting, where MySQL is available for use by external applications, for example: https://www.hetzner.com/webhosting

By default, your project uses SQLite DBMS. How to switch your project to using MySQL is shown in the figure below:



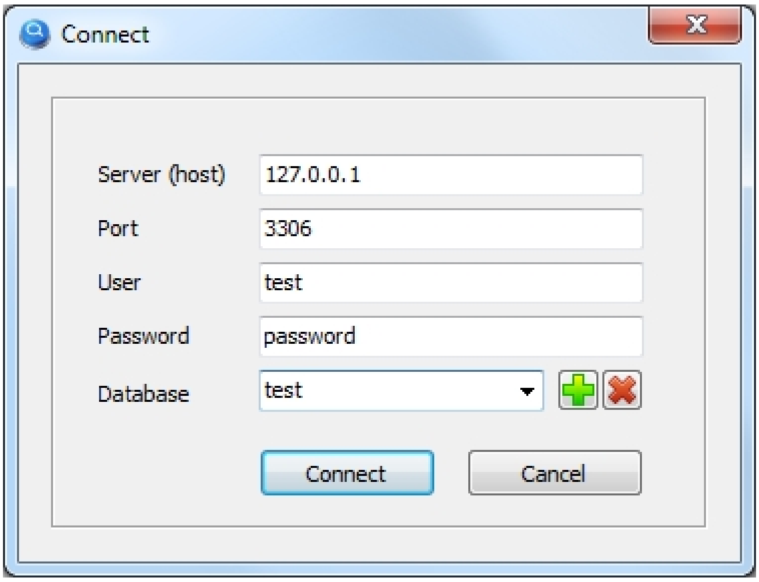
Using MySQL RDBMS is applicable for creating client–server applications operational via Internet or Intranet (Local Network).

It is required additional installation of MySQL Server.

This database is recommended for advanced users/developers.

http://www.mysql.com

To connect your project to MySQL DBMS, you need to specify such data as: server address (Server), port (Port), user name (User), password (Password) and database name (Database).



Server (host) 127.0.0.1

Port 3306

User test

Password password

Database test

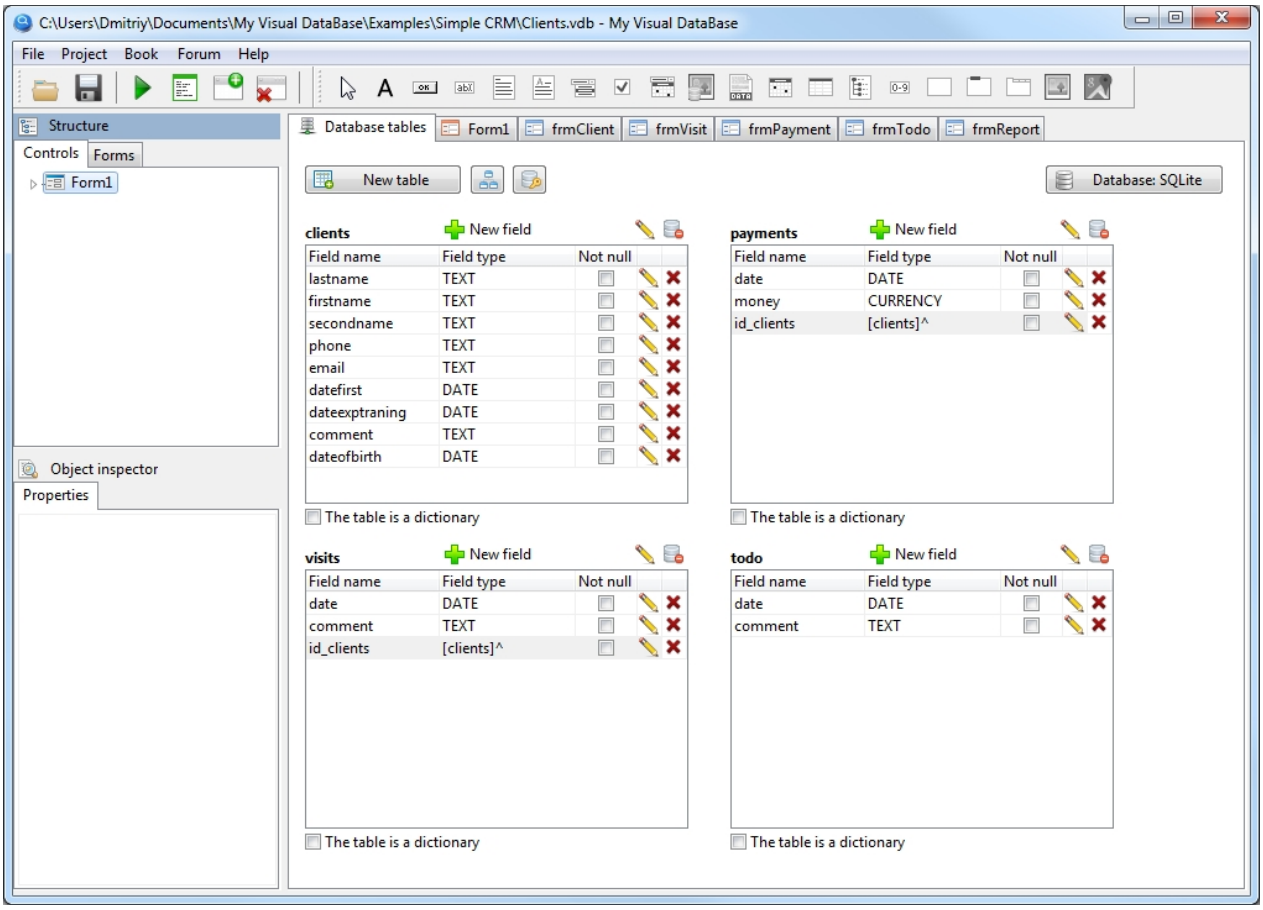
This documentation does not cover issues related to installation and configuration of MySQL DBMS, as its use implies that you already have basic skills for its use. To get the initial skills for working with this DBMS, you can easily find many sources on the Internet, such as the search query "intro to mysql".

Database design

Introduction

Important! *If you are not familiar with the basic principles of database structure design, please read the material on this link, namely, the first chapter "1.Introduction".*

Creation of the database structure of your application is performed on the "Database tables" tab.



clients

Field name	Field type	Not null
lastname	TEXT	□
firstname	TEXT	□
secondname	TEXT	□
phone	TEXT	□
email	TEXT	□
datefirst	DATE	□
dateexptraning	DATE	□
comment	TEXT	□
dateofbirth	DATE	□

□ The table is a dictionary

payments

Field name	Field type	Not null
date	DATE	□
money	CURRENCY	□
id_clients	[clients]^	□

□ The table is a dictionary

visits

Field name	Field type	Not null
date	DATE	□
comment	TEXT	□
id_clients	[clients]^	□

□ The table is a dictionary

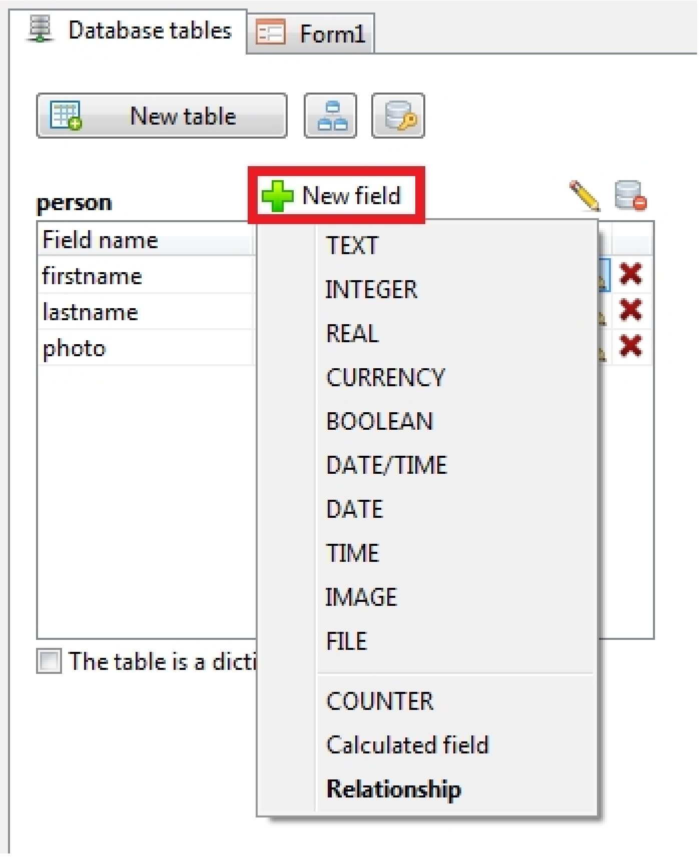
todo

Field name	Field type	Not null
date	DATE	□
comment	TEXT	□

□ The table is a dictionary

The database structure consists of tables and fields. To create a table, click the "New table" button. Click the "New field" button to create a field in the table.

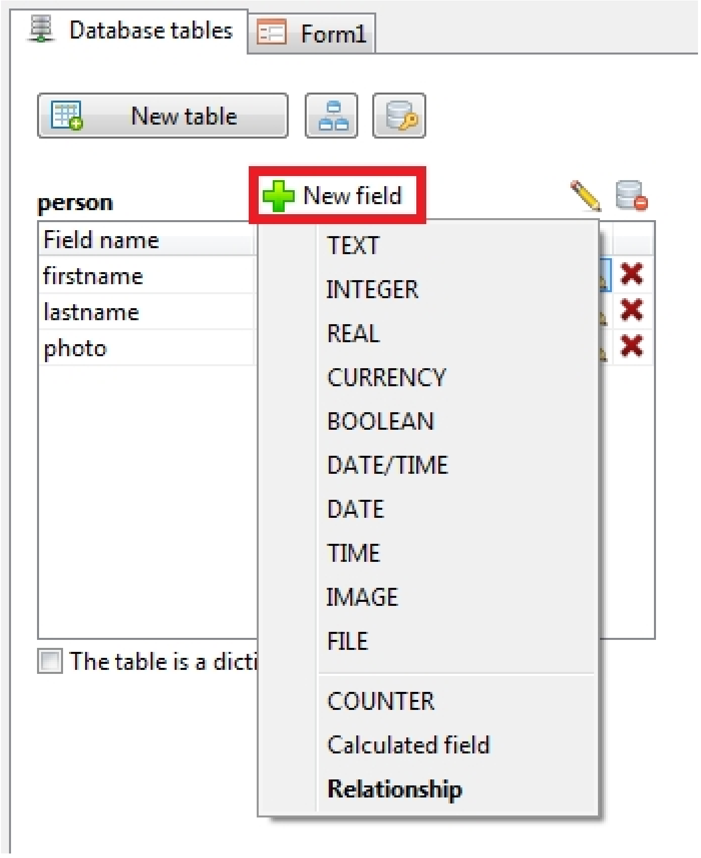
When you create a new field in the table, you must select its type.



More information about field types can be found in the "Data types" section.

Data types

When you create a new field in the table, you must select its type.

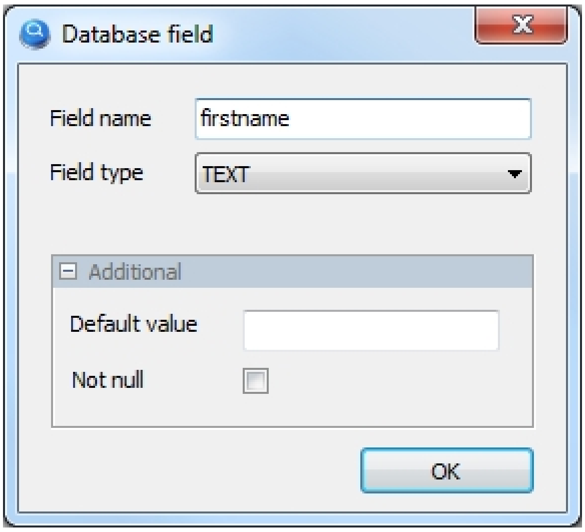


The following types are available:

Data type	Description
TEXT	any text, such as name or company name
INTEGER	number without fractional part, e.g. the number of something in pieces
REAL	floating-point number, such as 3.14
CURRENCY	is the same as the "REAL" type, but allows you to specify the formatting, for example: $25.00
BOOLEAN	assumes Yes or no value
DATE/TIME	date and time, for example 12/31/2020 11:00:00AM
DATE	date only
TIME	time only
IMAGE	type allows you to save the image as directly in the database or save a link to an external ima
FILE	type allows you to save the file as directly in the database or save a link to an external file on
COUNTER	automatically assigned a unique sequential value for the record
Calculated field	calculation field. Other table fields, SQL sub-queries and built-in functions can be used as arg
Relationship	special field, for establishing links between tables

Depending on the data type, different settings are available for the field to be created.

Adjustment of fields with type: TEXT, INTEGER, REAL, BOOLEN, DATE/TIME, DATE, TIME, IMAGE, FILE



Database field

Field name: firstname

Field type: TEXT

Additional

Default value:

Not null □

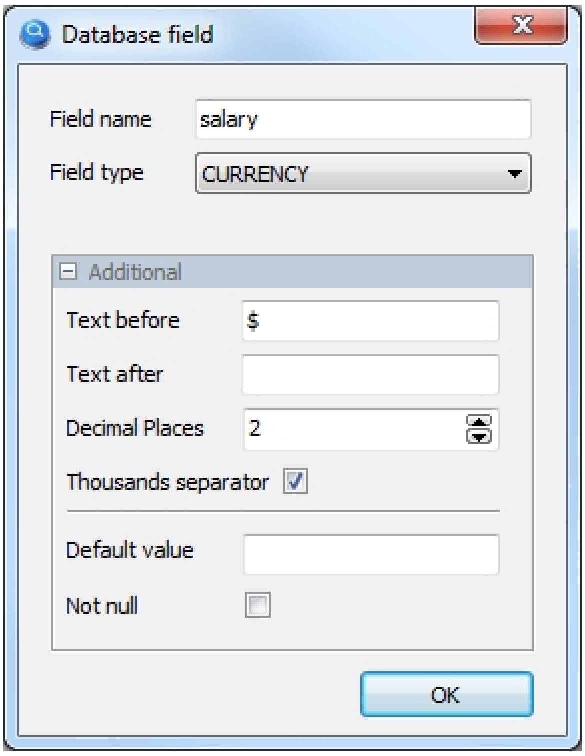
For the fields of these types, you can specify

Default value:
This value will be added to all records for this field, unless another value is specified.

For fields with DATA/TIME type, the following date-time format is used: YYYY-MM-DD HH:MM:SS, e.g.: 2020-01-31 12:00:00
For fields with DATA type, the following format is used: YYYY-MM-DD, e.g.: 2020-01-31
For fields with TIME type, the following format is used: HH:MM:SS, e.g.: 12:00:00

Not null:
This field will be mandatory. When creating and editing a record, if the value for this field is left blank, the user will be notified of the necessity to fill it.

Setting up a field with the type: CURRENCY



Field name: salary

Field type: CURRENCY

⊟ Additional

Text before: $

Text after:

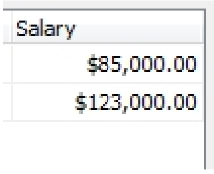
Decimal Places: 2

Thousands separator ☑

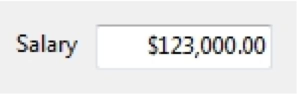
Default value:

Not null ☐

These settings allow you to set the currency format, such as unit designation, number of decimal places and thousand separator. This way, the currency values in TableGrid and Edit components will look like this:



Salary
$85,000.00
$123,000.00

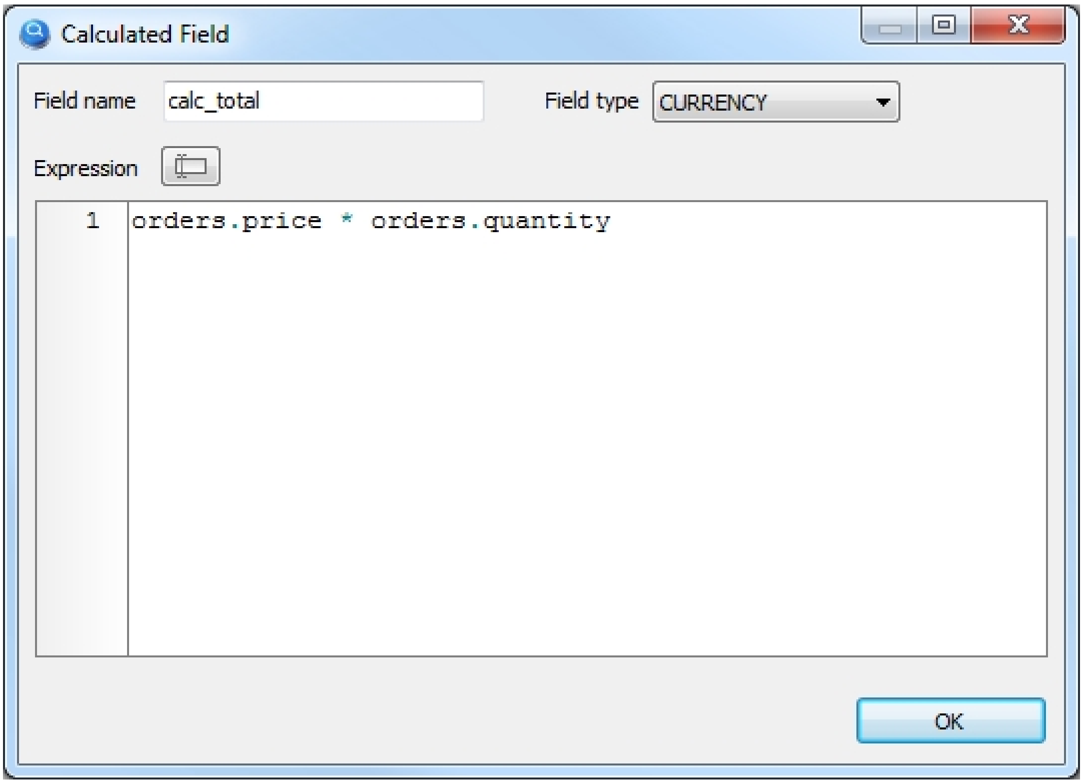


Salary: $123,000.00

Calculated field

A special field type created as a result of calculations based on existing fields in the table.

For example, you have such fields as "price" and "quantity" to find out the full cost of the order, you need to multiply the "price" by "quantity", you can do this using the calculated field.

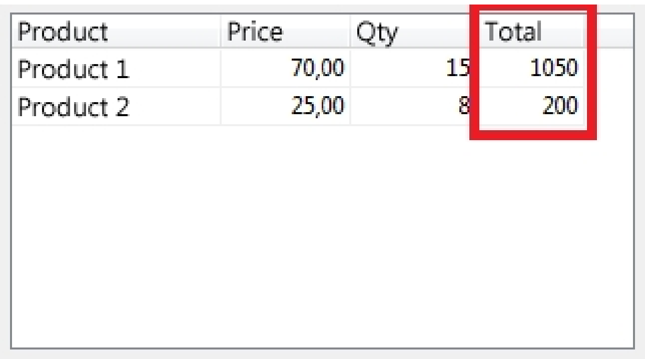


Field name: calc_total

Field type: CURRENCY

```
orders.price * orders.quantity
```

This way, you can see the result of the calculation in the TableGrid component as a usual column.

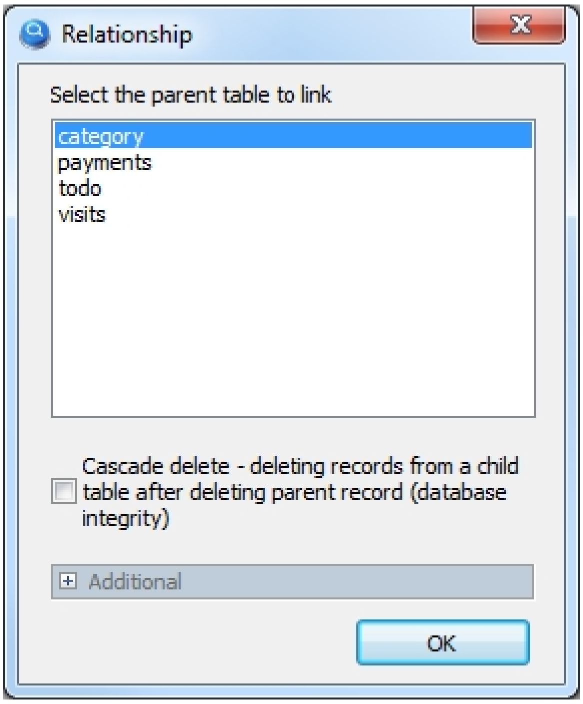


Product	Price	Qty	Total
Product 1	70,00	15	1050
Product 2	25,00	8	200

Also in the calculated field you can write an SQL query, which must be enclosed in **brackets**.

Relationship

To create external keys to other database tables, a special field type is used.



Select the parent table to link

- category
- payments
- todo
- visits

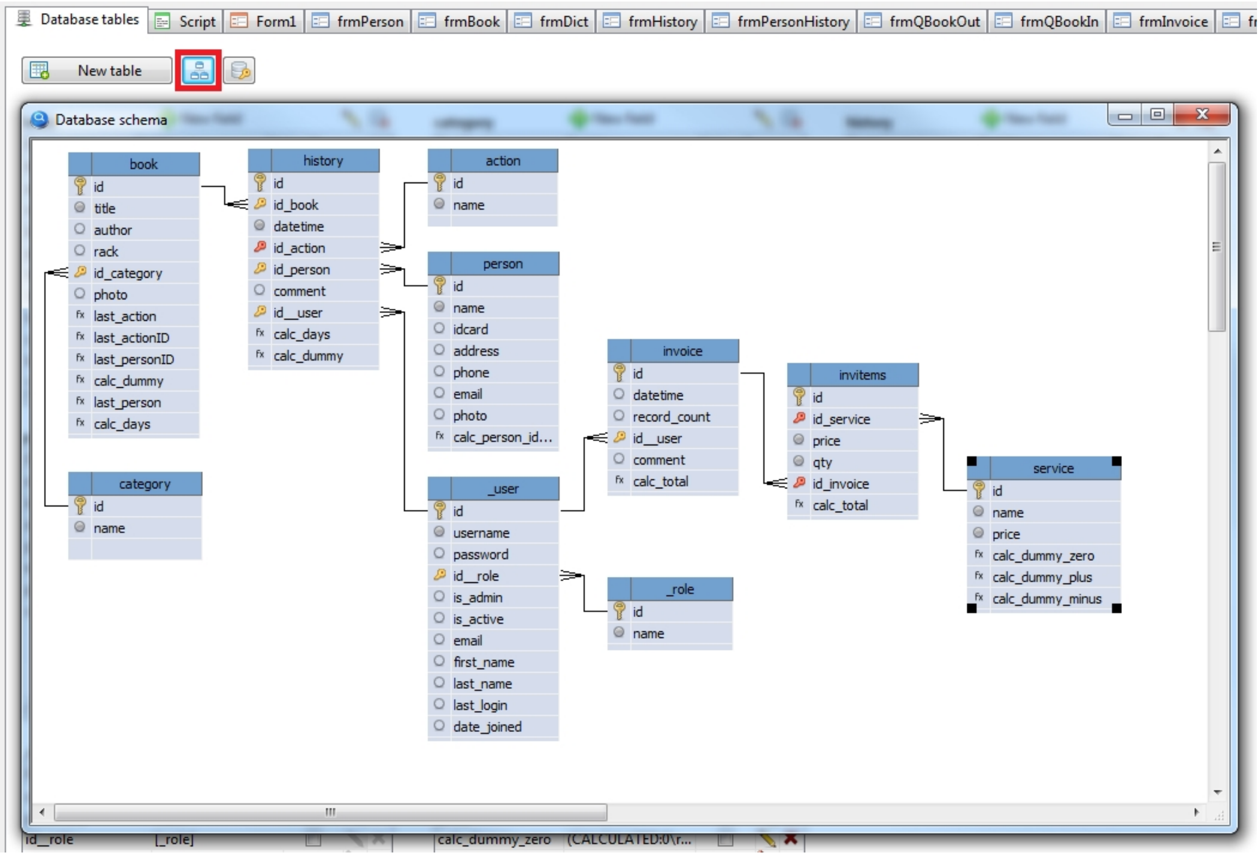
□ Cascade delete - deleting records from a child table after deleting parent record (database integrity)

Pay attention to the checkbox "Cascade delete". This option is necessary to support data integrity, for example, if you delete a client from the database, then all orders, which belong to this client, will be automatically deleted.

Important! *If you are not familiar with the basic principles of database structure design, please read the material on this link, namely, the first chapter "1.Introduction".*

Database schema

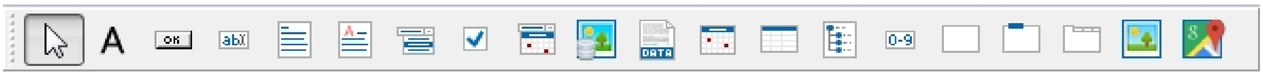
If your database already has many tables, it is easy to get confused in its structure. To avoid it, you can always visualize its graphical representation.



User interface designer

Introduction

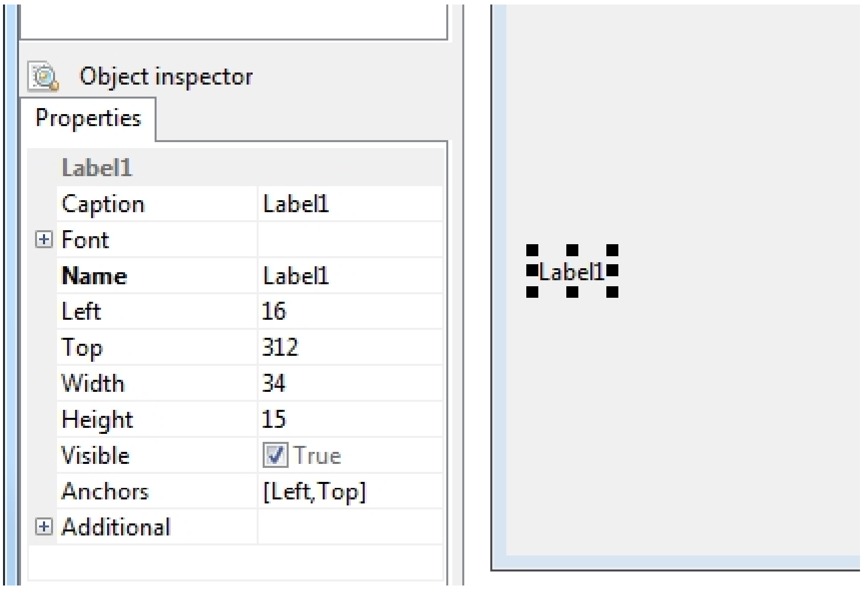
To create an user interface, you have access to components that you can see on the toolbar.



	Name	Description
A	**Label**	Label is a control that displays text on a form. Use Label to add text that the user can't edit to
OK	**Button**	Button is a push button control. Use Button to put a standard push button on a form. An impo to save a record to the database.
ab|	**TextBox**	It is used to enter numerical and text information. TextBox controls can also display text to the
	Memo	Multiline edit boxes allow the user to enter more than one line of text.
	RichEdit	Rich text edit controls let the user enter text that includes variation in font attributes, paragrap
	ComboBox	Allows you to select a value from a list.

	CheckBox	CheckBox represents a check box that can be on (checked) or off (unchecked). The user can c option.
	DateTimePicker	DateTimePicker is designed specifically for entering dates or/and times.
	DBImage	Allows saving the image to a database.
	DBFile	Allows saving a file to a database.
	Calendar	Calendar is a component that displays the month calendar of the specified year.
	TableGrid	Shows the database entries as a table.
	TreeView	It serves for output and creation of data in a hierarchical form (tree structure).
	Counter	Allows you to assign a unique number to records.
	Panel	Decorative interface element. It is a container for other components.
	GroupBox	Decorative interface element. It is a container for other components.
	PageControl	PageControl is a set of pages used to make a multiple page dialog box.
	Image	Use Image to display a graphical image on a form.
	Map	Allows you to place an interactive geographical map of Google Maps on the form, with the al

Each component has many properties that allow you to customize it to your needs. You can change the properties of a component in the "Object inspector" panel:



Object inspector

Properties

Label1	
Caption	Label1
⊞ Font	
Name	Label1
Left	16
Top	312
Width	34
Height	15
Visible	☑ True
Anchors	[Left,Top]
⊞ Additional	

For convenience, the properties of all components are divided into two types: basic and additional. Basic properties are available immediately when you select a component. To access additional properties, open the Additional section, which is located at the very end of the list of properties. As a rule, Additional properties are used less often than Basic properties.

Button actions

Introduction

The button is an important visual component. You will need to select the action for the button, which will be performed when the user clicks on it. There are 11 actions to choose from:

- Search
- New record
- Save record
- Show record
- Delete record
- SQL query
- Report
- Report (SQL)
- Show form
- Close form
- Open in Excel

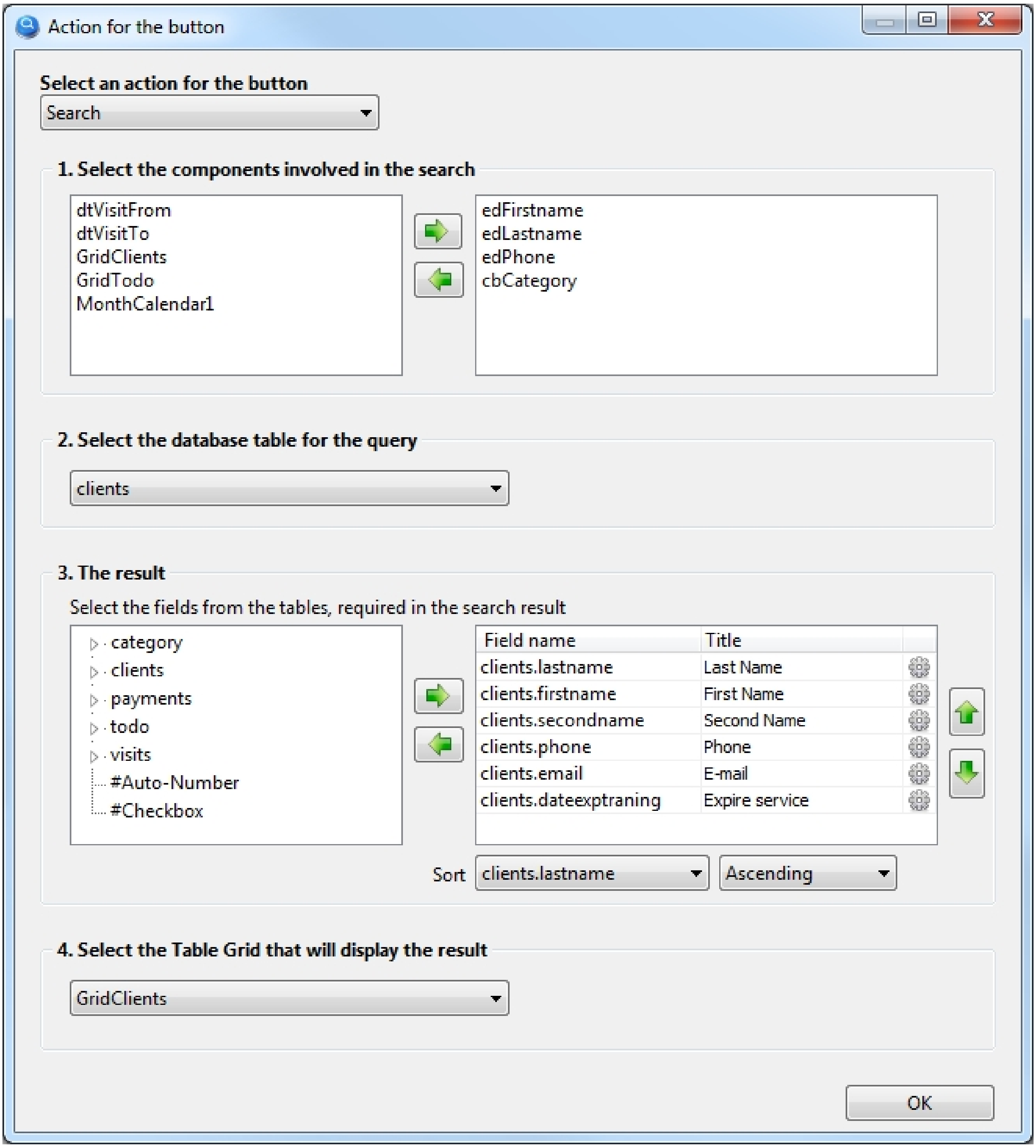
In addition, a button can be linked to a script that will also be executed when you click on it, you can read more about this in the Script section.

Search

Description

Allows you to configure the database search and display the search result in the selected TableGrid component.

To set up a database search, you must set up the button consisting of 4 steps.



Action for the button

Select an action for the button

Search

1. Select the components involved in the search

dtVisitFrom
dtVisitTo
GridClients
GridTodo
MonthCalendar1

edFirstname
edLastname
edPhone
cbCategory

2. Select the database table for the query

clients

3. The result

Select the fields from the tables, required in the search result

- category
- clients
- payments
- todo
- visits
- #Auto-Number
- #Checkbox

Field name	Title
clients.lastname	Last Name
clients.firstname	First Name
clients.secondname	Second Name
clients.phone	Phone
clients.email	E-mail
clients.dateexptraning	Expire service

Sort clients.lastname Ascending

4. Select the Table Grid that will display the result

GridClients

1. Select the names of the components whose contents will participate in the search.
   I.e. we select the components in which we will enter the search criteria, it can be textboxe, a combobox, a date selection component, etc.
   In this example, we have set up a search by client's Surname, Name, phone number and category.
   Don't forget! All components in this list should have TableName and FieldName properties filled in.

2. Select the database table in which we will search for information.
   In this example, we are looking for clients, so we select a database table with the name *client*

3. Choose which fields in the database table we need as a result of the search.

We also give the names of the headers for the columns with the search result.
In the third column, you can specify a formula to calculate the total value in the footer and set the alignment. More info.
If necessary, you can choose by which database field to sort the search result.

Pay attention to the line on the left side: **#Auto-Number**, you can use it to add a sequential numbering to the search result. Using the **#Checkbox** line, you can add a checkbox column to the search result to mark the necessary entries. This column is usually used in conjunction with the script.

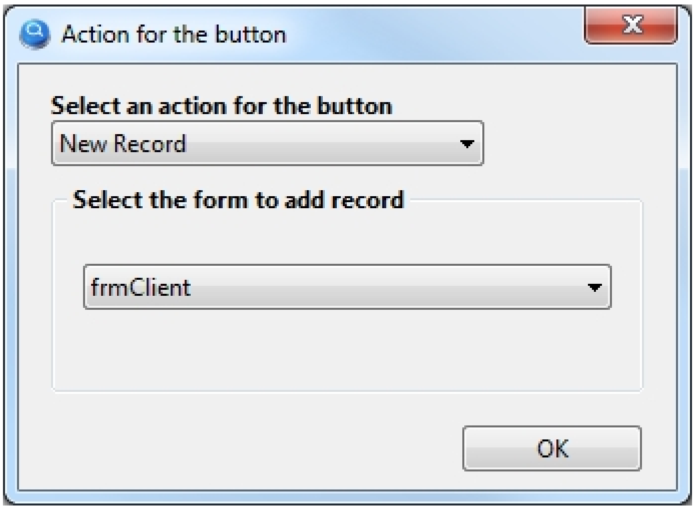
4. Select the TableGrid component where the search result will be displayed.

New record

Description

It is used to call up a form that will be used to create a new record.

It is necessary to choose a form from the list that will be used to create a new record. Thus, when you click on this button, the selected form will be shown on the screen.



Action for the button

Select an action for the button: New Record

Select the form to add record: frmClient

The selected form must have a button with the action "Save record", otherwise you can not create a new record in the database using this form.

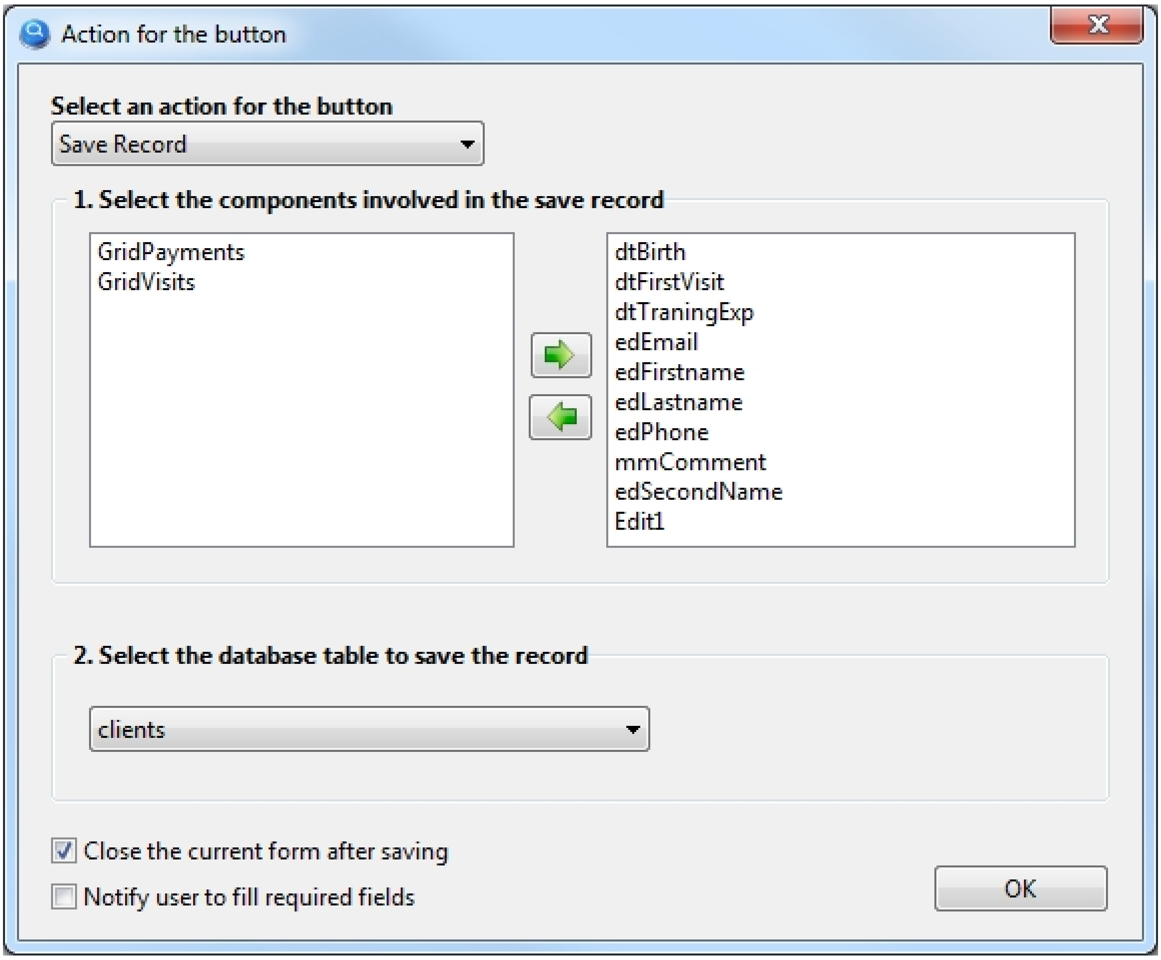
Important! *Usually the form for adding and editing a record is the same.*

Important! *Sometimes users choose "Show form" instead of "New record", this is an error, do not confuse, because this action not only shows the form, but also prepares it to create a new record!*

Save record

Description

Saving information from a form to a database.



Action for the button

Select an action for the button: Save Record

1. Select the components involved in the save record

GridPayments
GridVisits

dtBirth
dtFirstVisit
dtTraningExp
edEmail
edFirstname
edLastname
edPhone
mmComment
edSecondName
Edit1

2. Select the database table to save the record: clients

☑ Close the current form after saving

□ Notify user to fill required fields

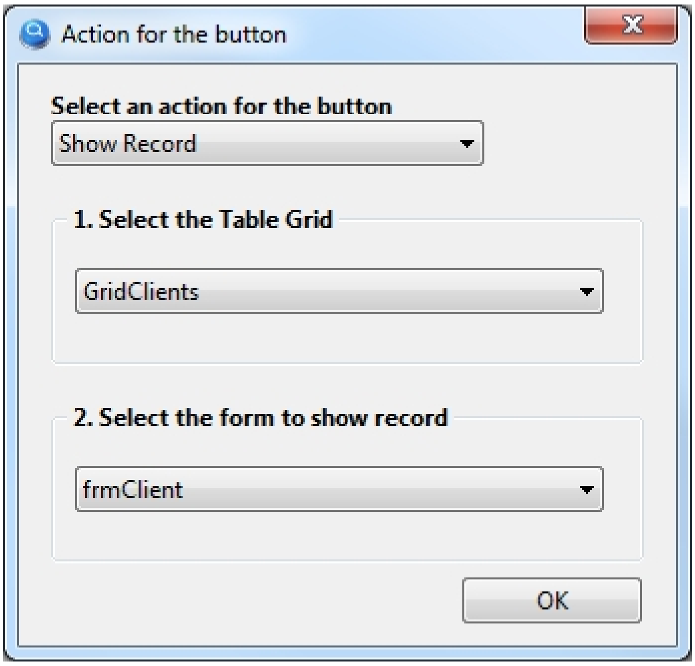
1. Select the components, the information from which will be saved to the database.
   Important! *All components in this list should have the TableName and FieldName properties filled.*

2. Select the database table to which we will save the information.
   In this example we are saving information about the client, so we select the "clients" table.

Show record

Description

It is used to call a form intended for editing (viewing) a record.



Action for the button

Select an action for the button: Show Record

1. Select the Table Grid: GridClients

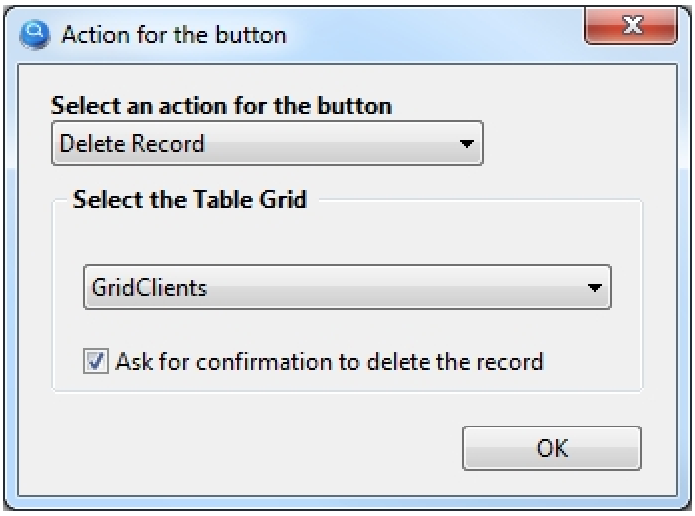
2. Select the form to show record: frmClient

1. Select the TableGrid component where we will select the record to edit (view).
2. Choose the form, which is intended for editing (viewing) the record.
   Important! *Usually the form for adding and editing a record is the same.*

Delete record

Description

Delete the selected record from the database.



Action for the button

Select an action for the button: Delete Record

Select the Table Grid: GridClients

☑ Ask for confirmation to delete the record

Select the TableGrid component in which we will select the record to be removed from the database.

SQL query

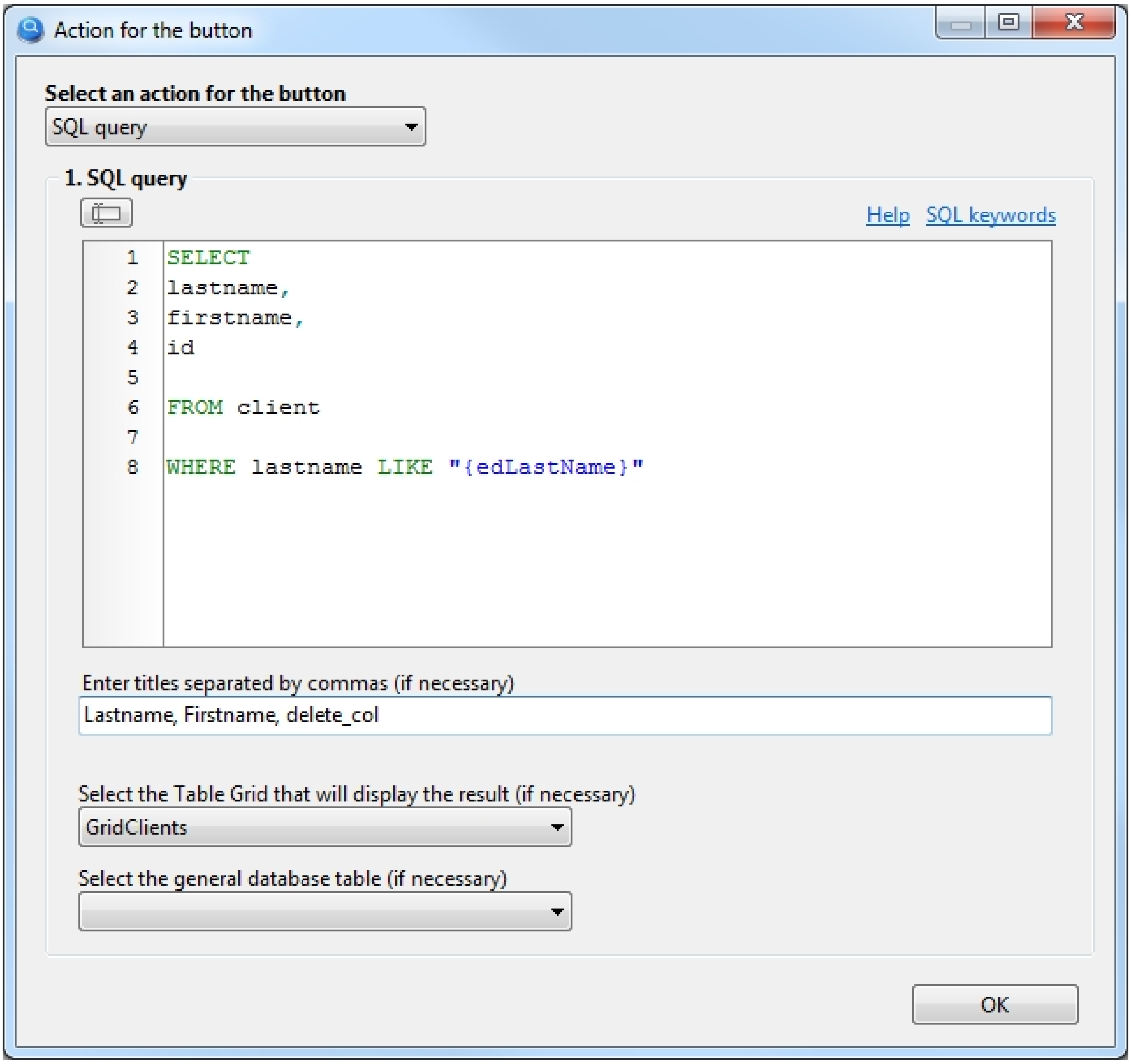
Description

Allows you to enter an SQL query for execution and, if necessary, display the result of the query in a TableGrid component.

Important! *To work with this action, you must have knowledge of the SQL query language.*

For SQL query you can also use UPDATE, INSERT, DELETE and any other keywords in the syntax.

Let's consider an example of an SQL query to search for records with the desired last name.



```
SELECT
lastname,
firstname,
id

FROM client

WHERE lastname LIKE "{edLastName}"
```

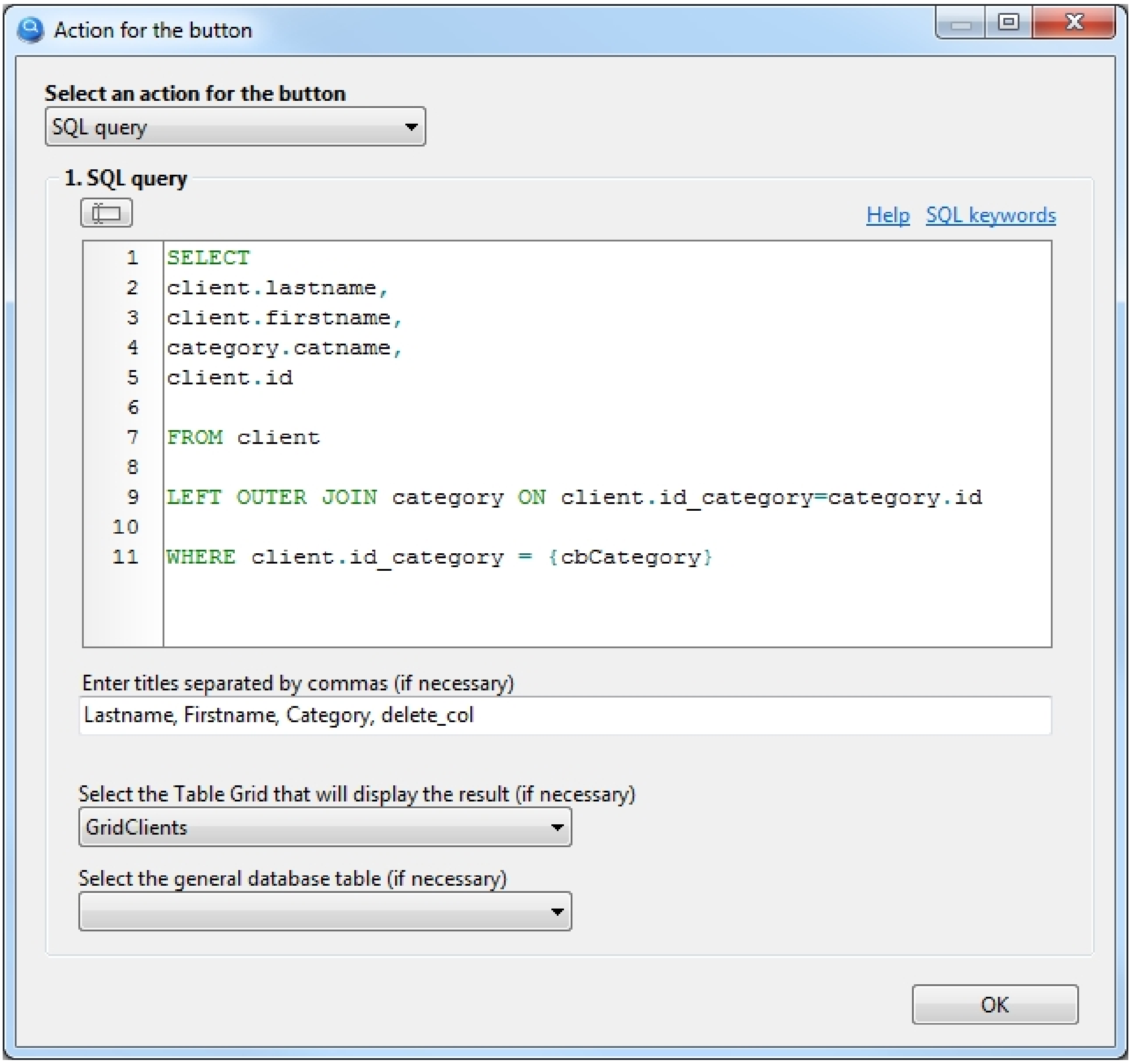
Enter titles separated by commas (if necessary): Lastname, Firstname, delete_col

Select the Table Grid that will display the result (if necessary): GridClients

Select the general database table (if necessary):

Notice in the SQL query the text of **{edLastName}**, where edLastName is the name of the component (TextBox) in the current form, so the text from this component will be automatically inserted into the SQL query.

Let's consider one more example where we search for clients by their category.



Select an action for the button: SQL query

1. SQL query

```
SELECT
client.lastname,
client.firstname,
category.catname,
client.id

FROM client

LEFT OUTER JOIN category ON client.id_category=category.id

WHERE client.id_category = {cbCategory}
```

Enter titles separated by commas (if necessary): Lastname, Firstname, Category, delete_col

Select the Table Grid that will display the result (if necessary): GridClients

Select the general database table (if necessary):

Note in the SQL query for the text of {cbCategory}, where cbCategory is the name of the component (ComboBox) in the current form, so the SQL query will automatically insert the identifier of the category that is selected in the cbCategory component. Thus, will be found clients who belong to the category specified in the cbCategory component.

In all these examples, in the query we added **id** (e.g.: SELECT lastname, fistname, **id** FROM...), which is mandatory if we want to be able to edit or delete a record from a table component, in our case with the name GridClients.

If you do not want to see the id value in a TableGrid component, enter a name for this column delete_col

You can use the keyword "$autoinc" to add sequence numbering to the TableGrid component where the query result will be output. For example: SELECT "$autoinc", somefield FROM table

Please note the option: "Select the general database table". In case of complex SQL queries with sub-queries, you need to select the main database table yourself.

In cases when the TableGrid component was populated with data using SQL query, this component will not be automatically updated when data changes in the database table. You can update data in the component as follows: if Form1.TableGrid1.dbSQL <> '' then Form1.TableGrid1.dbSQLExecute;

Report

Description

Allows printing data from a database.

- How to print a record
- How to print a simple list
- How to print a master-detail report
- How to print a mater-detail report with grouping

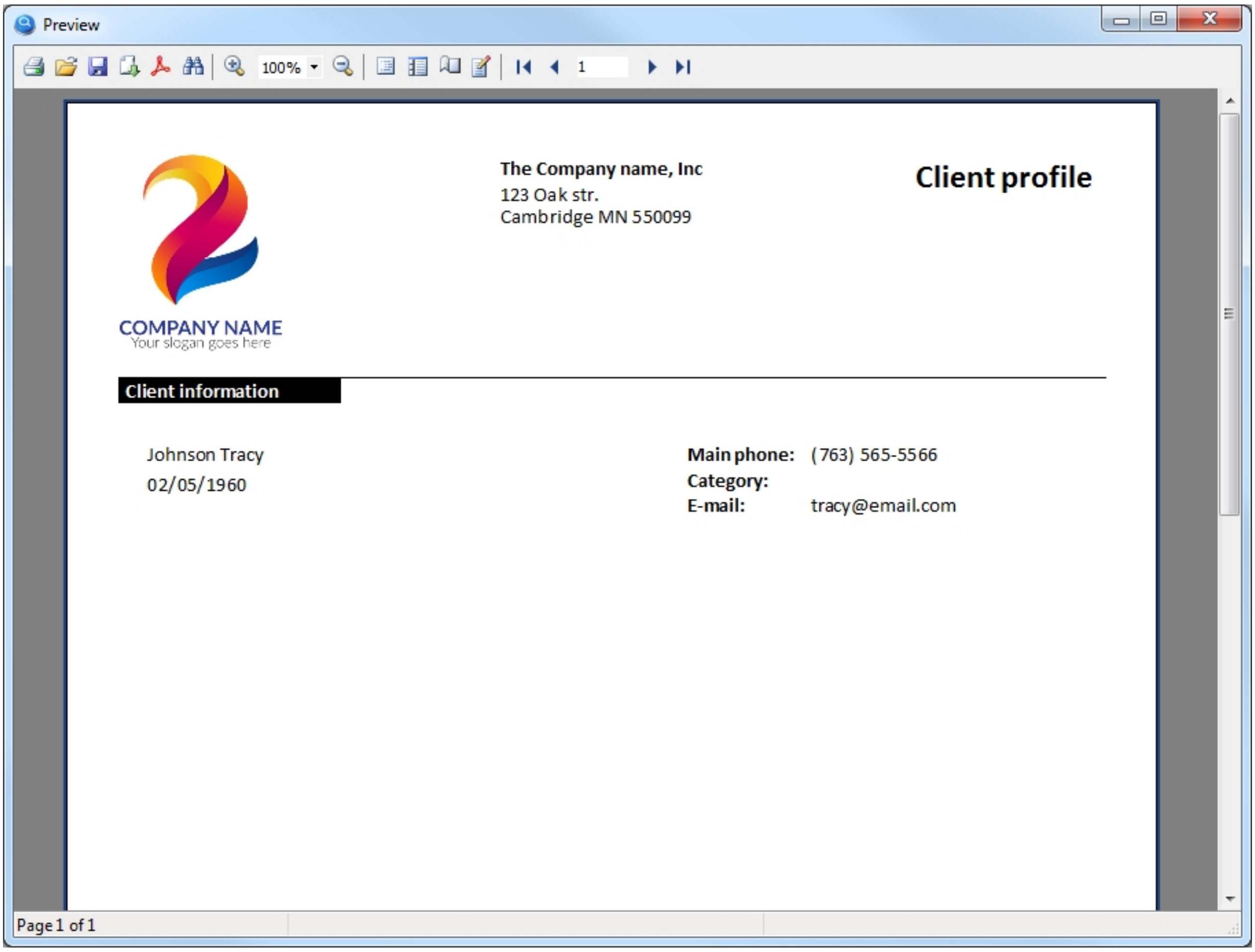
How to print a record

Description

Allows printing data from a database.

The setting of the Report button is almost the same as the setting of the Search button, since in fact both actions search the database and then display the search result, only in our case we will see the search result when printing. After setting up the button, you need to call the report designer to create a template, which will determine exactly how the document will look when printed, but everything in order.

As an example, we will print a simple client profile. It will look as shown in the picture:



The Company name, Inc
123 Oak str.
Cambridge MN 550099

Client profile

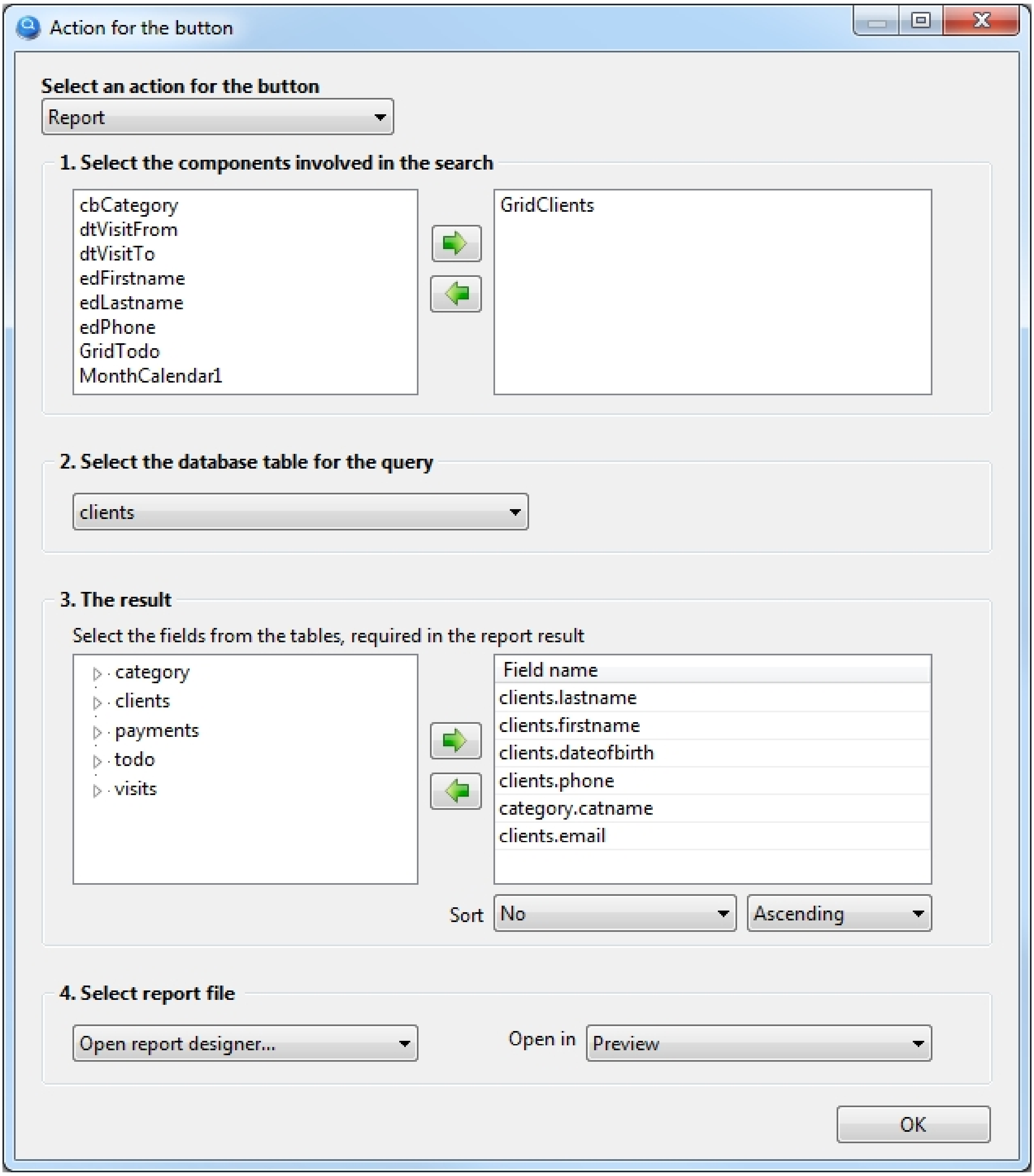
Client information

Johnson Tracy
02/05/1960

Main phone: (763) 565-5566
Category:
E-mail: tracy@email.com

Page 1 of 1

The setting of the "Report" button is shown in the figure below:



Select an action for the button: Report

1. Select the components involved in the search

cbCategory
dtVisitFrom
dtVisitTo
edFirstname
edLastname
edPhone
GridTodo
MonthCalendar1

GridClients

2. Select the database table for the query: clients

3. The result

Select the fields from the tables, required in the report result

category
clients
payments
todo
visits

Field name
clients.lastname
clients.firstname
clients.dateofbirth
clients.phone
category.catname
clients.email

Sort: No, Ascending

4. Select report file: Open report designer...

Open in: Preview

Let's consider this setting in detail by steps:

1. **Select the components involved in the search**
The GridClients component is selected here, where we see all our clients, so only the client that was selected in the GridClients component will be in the report. If there is no component in this list, then the report will receive all data from the selected database table.

2. **Select the database table for the query**
We will print information about the client, respectively select the "clients" database table.

3. **The result**

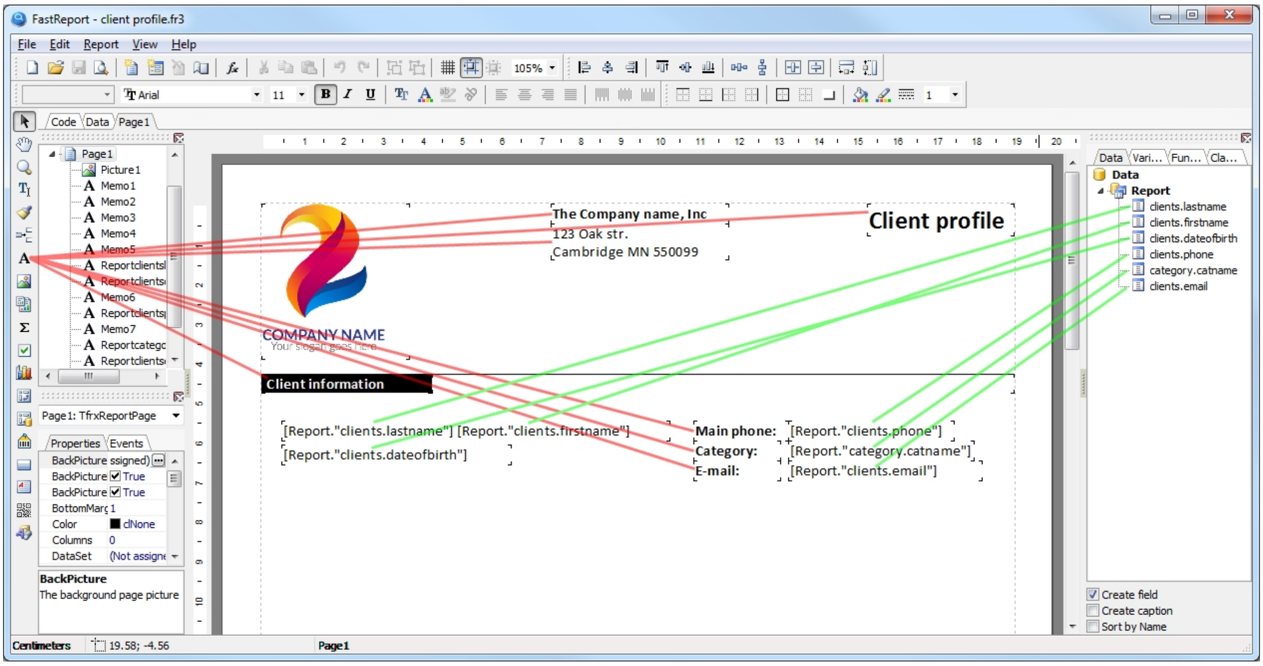
Select the table fields that we need in the report.

4. **Select report file**

Since the report template has yet to be created, this setting should select "Open report designer...".
A report template is a file that defines exactly what the report will look like, this file will be created in the report designer and will be saved in the Report folder of your project.

To create a report template, you need to run the project
In the launched project, click this button with the report action, so you will open the report designer.

There is nothing complicated about creating a simple template; in the figure below you can see where the objects were placed from and to. Do the same.



The Company name, Inc
123 Oak str.
Cambridge MN 550099

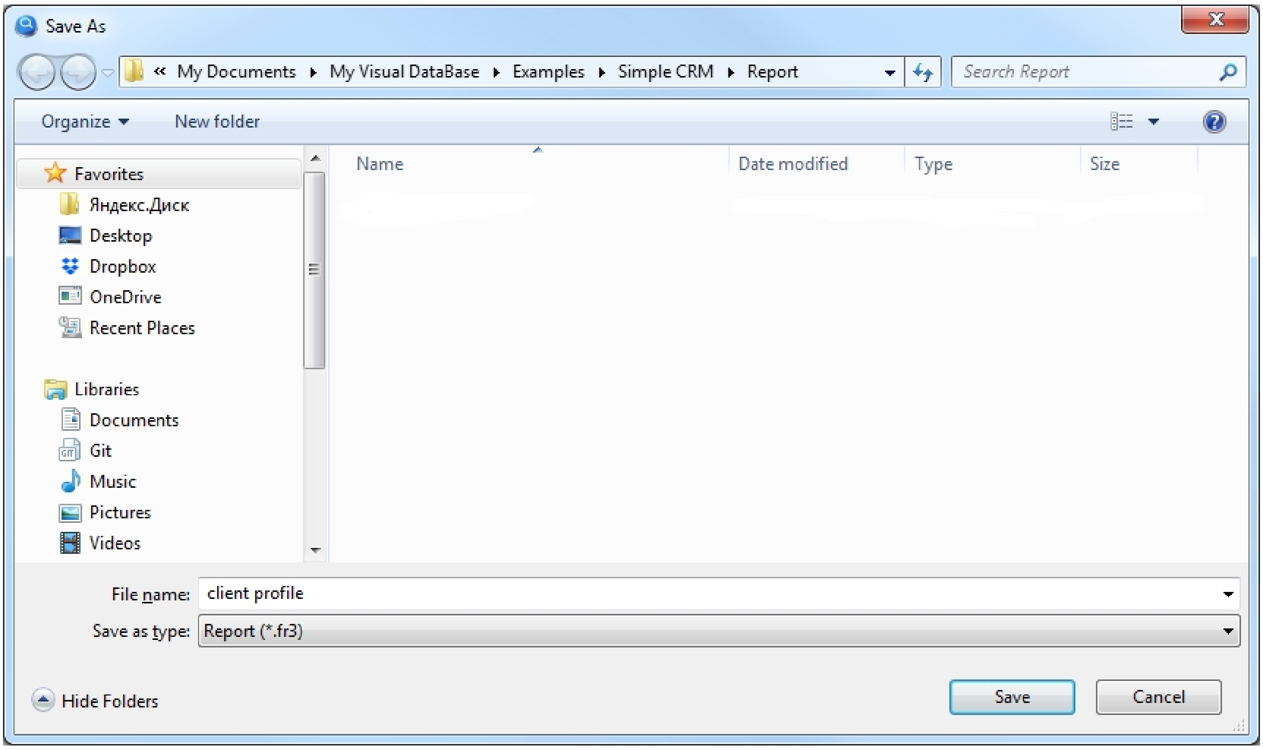
Client profile

Client information

[Report."clients.lastname"] [Report."clients.firstname"]
[Report."clients.dateofbirth"]

Main phone: [Report."clients.phone"]
Category: [Report."category.catname"]
E-mail: [Report."clients.email"]

If everything is ready, go to the File > Preview (Ctrl+P) menu and you should see the report ready to print, as was previously shown in the first picture in this section.
All you need to do is save the report file to the File > Save(Ctrl+S) menu. Name the report file, for example "client profile" as shown in the figure below:



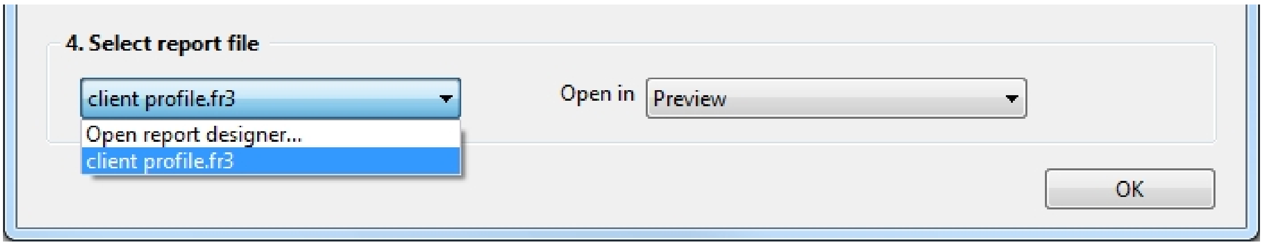
Save As

« My Documents ▸ My Visual DataBase ▸ Examples ▸ Simple CRM ▸ Report

File name: client profile

Save as type: Report (*.fr3)

The last step. Close the launched project and go back to the settings of this button.
In the settings item "4. Select a report template", instead of "Open report designer..." select the previously saved report template "client profile.fr3" from the list as shown in the figure below:



4. Select report file

client profile.fr3

Open report designer...
client profile.fr3

Open in Preview

Here we go. Now you can start the project again. After selecting the desired client in the GridClients component, click on this button and you will see a report with the selected client data ready for printing.

Sometimes it is convenient to place this button for printing directly on the form of adding/editing a record. In this case, leave the list blank in the button settings "**1. Select the components involved in the search**". If the button with the "Report" action is placed on the form intended for adding/editing a record, it automatically recognizes which one should be sent to print.

Documentation on working with the report designer can be found here https://www.fast-report.com/public_download/UserManual-en.pdf

Created with the Standard Edition of HelpNDoc: Full-featured multi-format Help generator

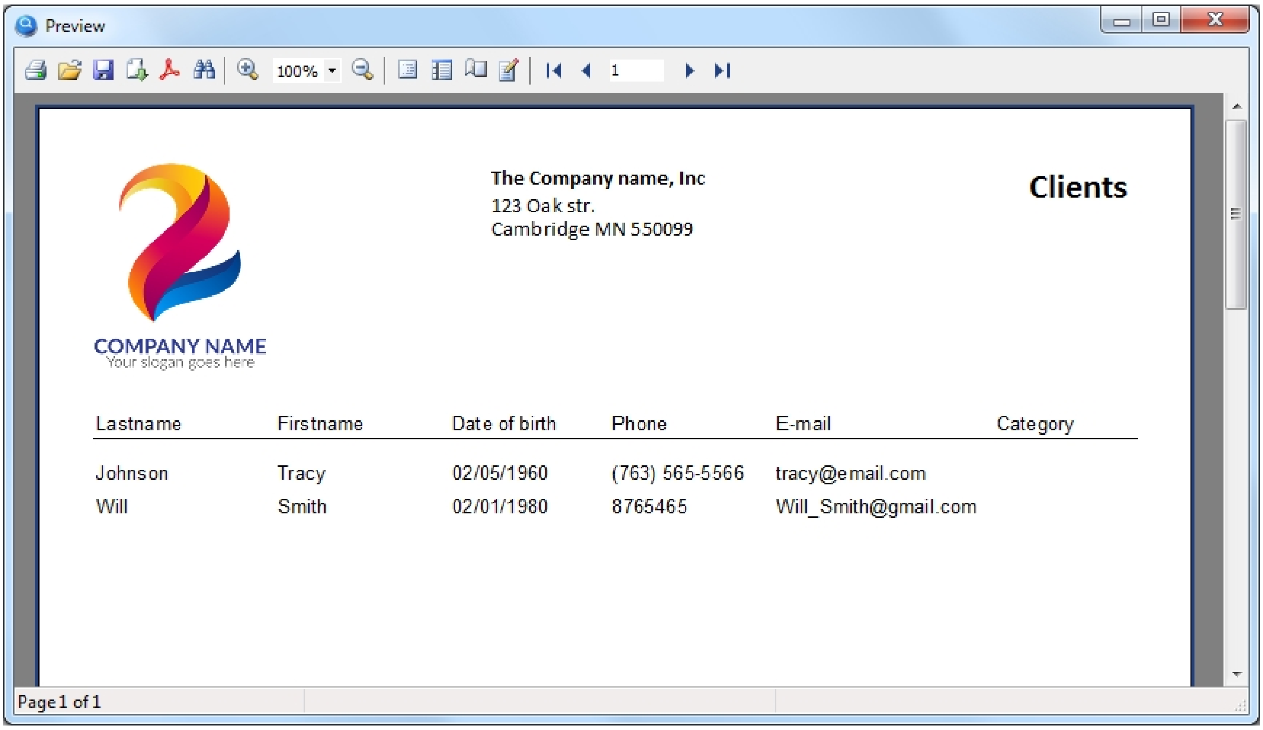
How to print a simple list

Description

Allows printing data from a database.

The setting of the Report button is almost the same as the setting of the Search button, since in fact both actions search the database and then display the search result, only in our case we will see the search result when printing. After setting up the button, you need to call the report designer to create a template, which will determine exactly how the document will look when printed, but everything in order.

As an example, we will print a simple list of clients. It will look as shown in the picture:

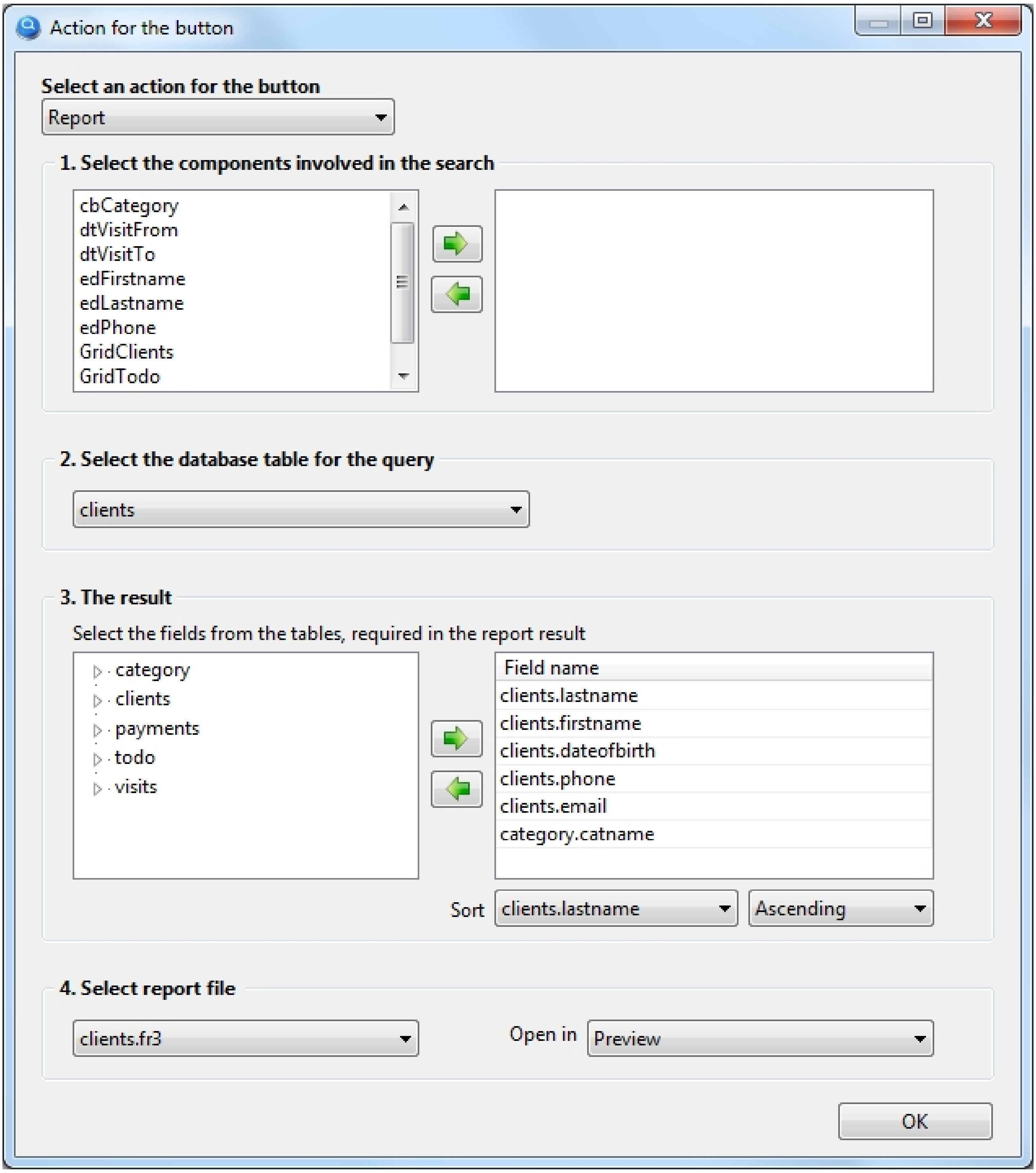


The Company name, Inc
123 Oak str.
Cambridge MN 550099

Clients

Lastname	Firstname	Date of birth	Phone	E-mail	Category
Johnson	Tracy	02/05/1960	(763) 565-5566	tracy@email.com	
Will	Smith	02/01/1980	8765465	Will_Smith@gmail.com	

The setting of the "Report" button is shown in the figure below:



Select an action for the button
Report

1. Select the components involved in the search

2. Select the database table for the query
clients

3. The result
Select the fields from the tables, required in the report result

Field name
clients.lastname
clients.firstname
clients.dateofbirth
clients.phone
clients.email
category.catname

Sort clients.lastname Ascending

4. Select report file
clients.fr3
Open in Preview

Let's consider this setting in detail by steps:

1. **Select the components involved in the search**
This list is empty, so the report will contain data on all clients that are present in the database. If necessary, you can add components that will only include clients that meet your criteria in the report.

2. **Select the database table for the query**
We will print information about the client, respectively select the "clients" database table.

3. **The result**

Select the table fields that we need in the report.

4. **Select report file**

Since the report template has yet to be created, this setting should select "Open report designer...".
A report template is a file that defines exactly what the report will look like, this file will be created in the report designer and will be saved in the Report folder of your project.
.

A nuance worth noting here. As you can see from the settings in p.2, to generate the result we select fields from 2 different tables (clients and category), as a rule, in other programs you need to write an SQL query, which manually specifies how one table, links to another table and in what sequence.

The program My Visual Database tries to understand by itself how tables should be linked to each other in order to get exactly the data you expect, thus saving you from the need to learn the SQL query language.

Unfortunately, it is not always possible for the program to predict how tables should be linked to get the data you would like to see. Such a situation can occur when you need to link 3 or more tables and if there are no obvious links between them.

What should we do? On the "Database tables" tab, under each table, there is a checkbox "The table is a dictionary". On this tab, you should note which tables in your project are dictionaries. But how to understand which tables are dictionaries?

Examples of dictionary tables are the table containing country names, Statuses (Open, Closed), Types (Legal Entity, Physical Entity), name with prices, etc.

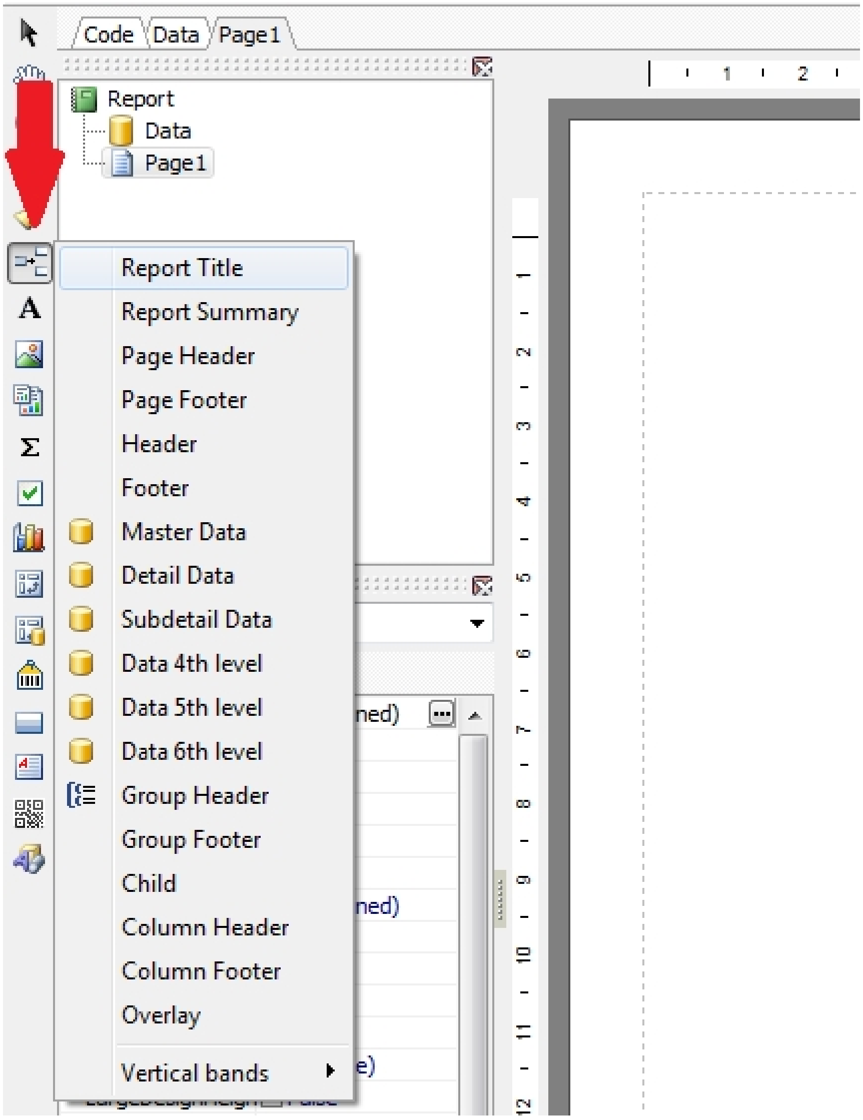
That is, such tables, which, as a rule, are filled in first at the start of work with the database and are not edited or edited rarely in the future.

In this case, the dictionary table can be referred to the "category" (client category), for this table you need to check the "Table is a dictionary" box under these tables, so you will help the program to link the tables correctly.

To create a report template, you need to run the project ▶
In the launched project, click this button with the report action, so you will open the report designer.

This report will be a bit more complicated than the previous one, as it will use so-called blocks (or Band). Blocks allow you to create almost any report structure. Some of them we will now get acquainted with.

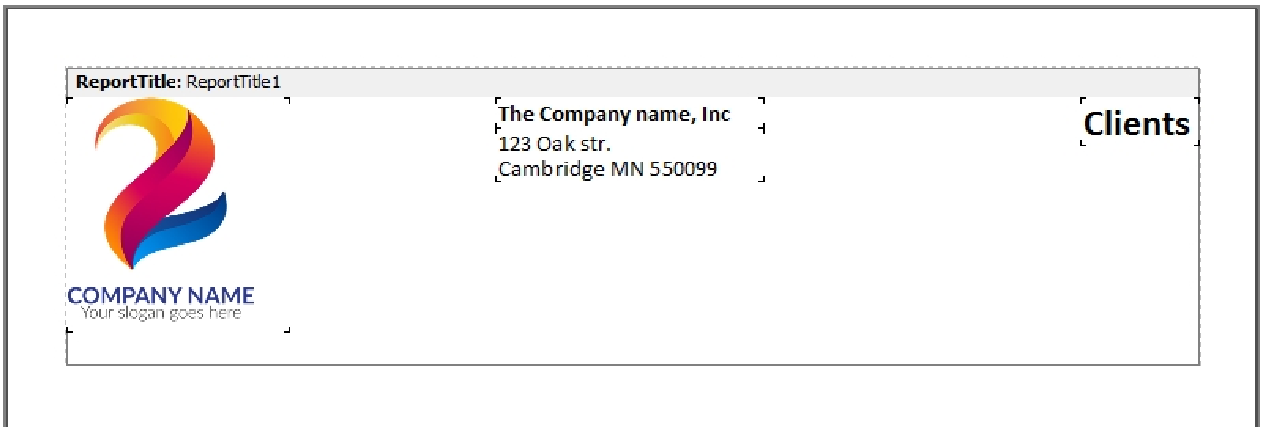
To view all available blocks, click on the left icon , then you will see a menu as shown in the figure below



Let's start creating a report using blocks and find out what they are for.

From the menu shown in the figure above, select the **Report Title** block and this block will appear in the report. The information in this block will be printed only on the first page of your report.

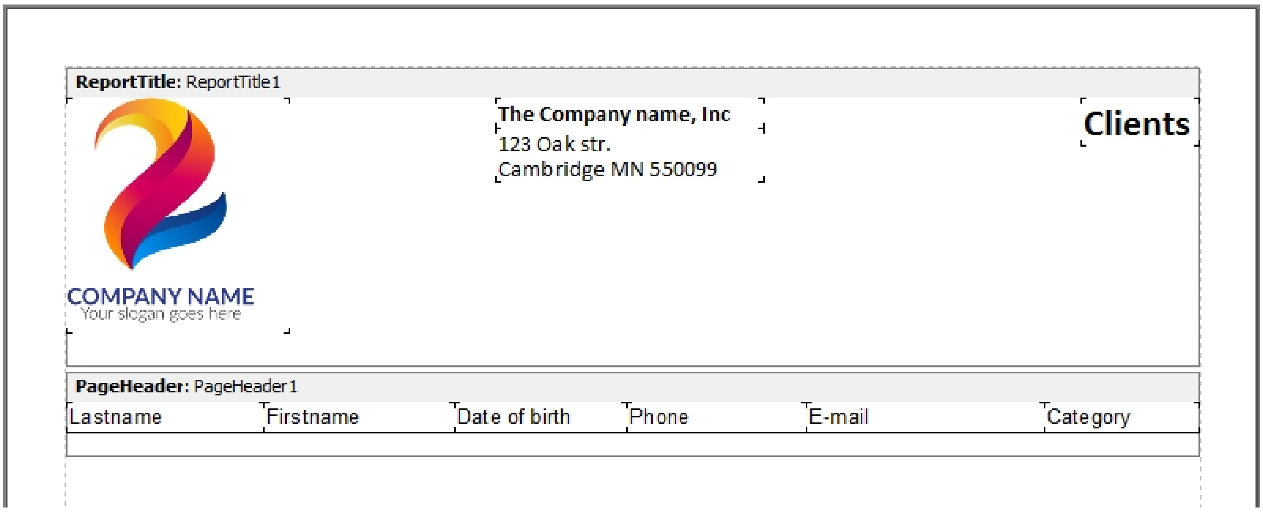
Place the logo and company information in this block. As a result, you should succeed as shown in the figure below:



You can skip this block if you do not need a title for your report.

Place the next **Page Header** block. The information placed in this block will be displayed on each printed page (in case your report does not fit on one page).

In this block we will place the information about the client. We will also place the headers for the table which will be located in the next block.
Place the text and data fields in this block as shown in the picture below:

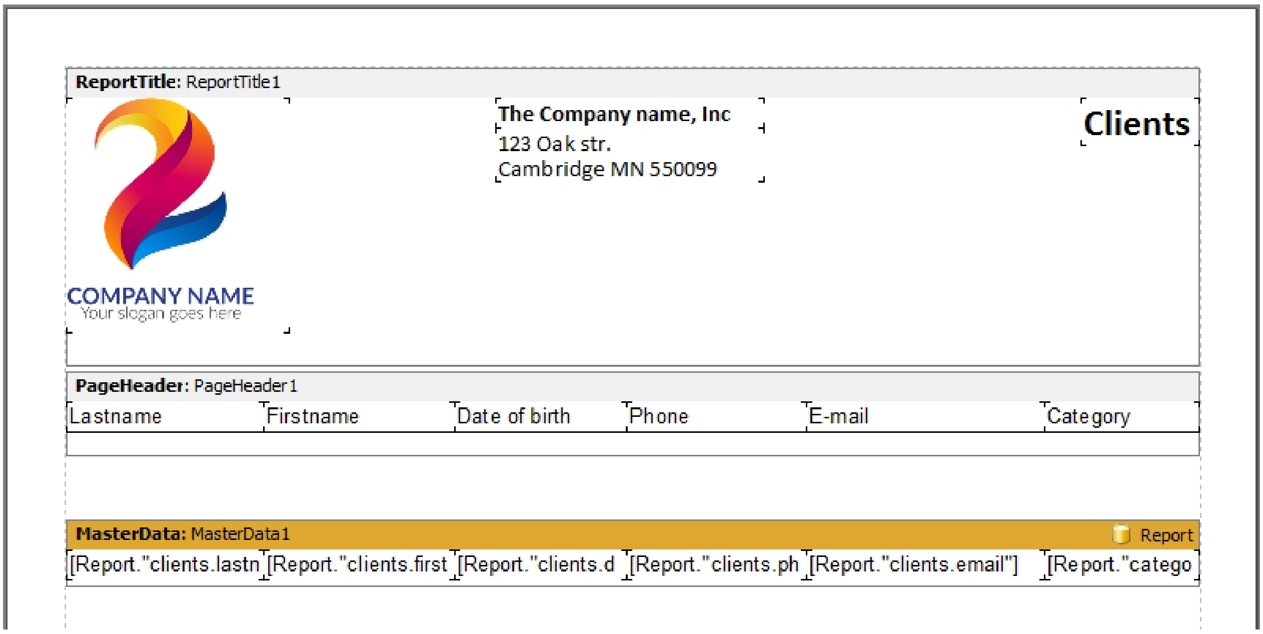


The next block to be placed in the report is **Master Data**.

This block is designed to output information in the form of a table. With this block we will get a table where we will see the list of clients.

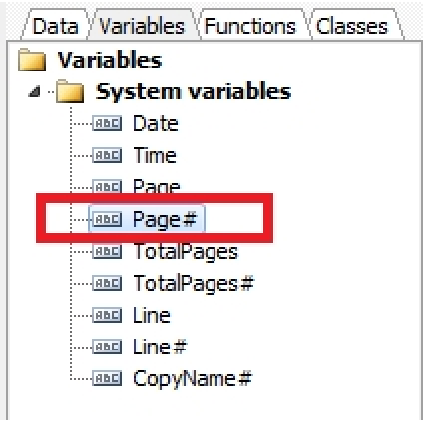
Place this block in the report by selecting it from the menu. Before it appears in the report, you will see a window with the title **Select DataSet**, in which you need to select the data source. Select a data source from the list with the title **Report** and click **OK**.

In this block, place the database fields that you need in the table. As a result, you will see, as shown in the figure below:

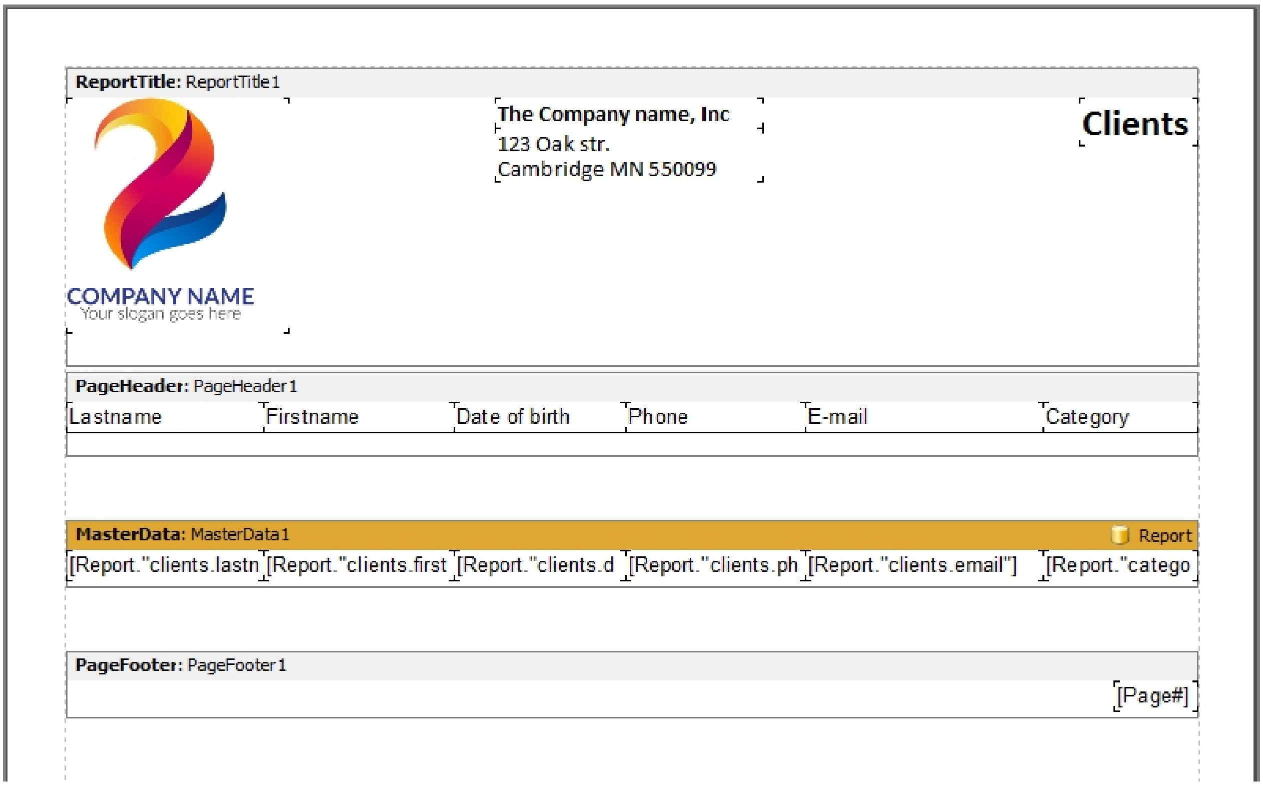


And the last block that we will place in the report is **PageFooter**, this block will be printed on each page of the report, we use it to numerate the pages.

Place the **Page#** system variable in this block from the right side of the report designer (Variables tab), just drag and drop:



Now your report template is fully ready



ReportTitle: ReportTitle1

The Company name, Inc
123 Oak str.
Cambridge MN 550099

Clients

PageHeader: PageHeader1

Lastname | Firstname | Date of birth | Phone | E-mail | Category

MasterData: MasterData1 — Report

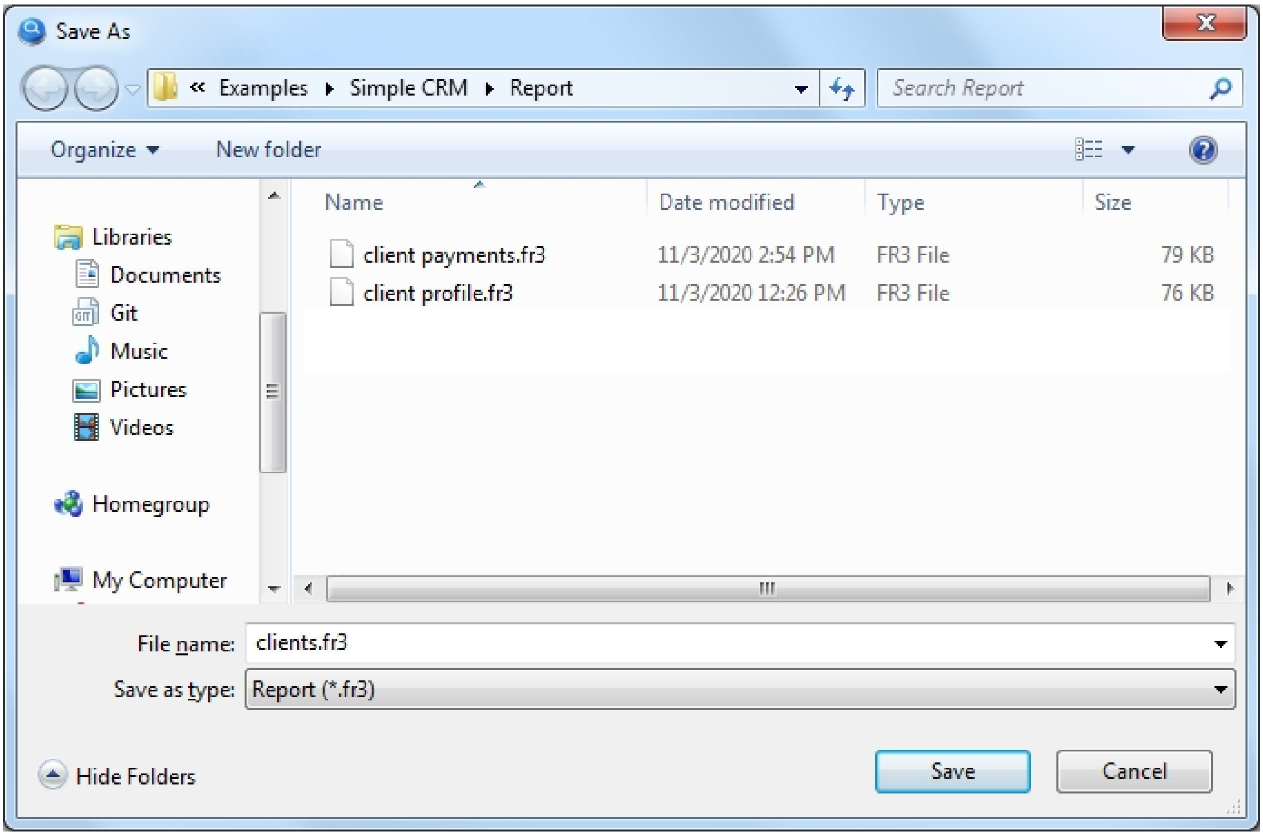
[Report."clients.lastn | [Report."clients.first | [Report."clients.d | [Report."clients.ph | [Report."clients.email"] | [Report."catego

PageFooter: PageFooter1

[Page#]

Go to the File > Preview menu or just press Ctrl+P to see what your report will look like.

Save the report template to the **Report** folder of your project, menu **File > Save As**... Give the file a name such as "**clients**" as shown in the figure below:

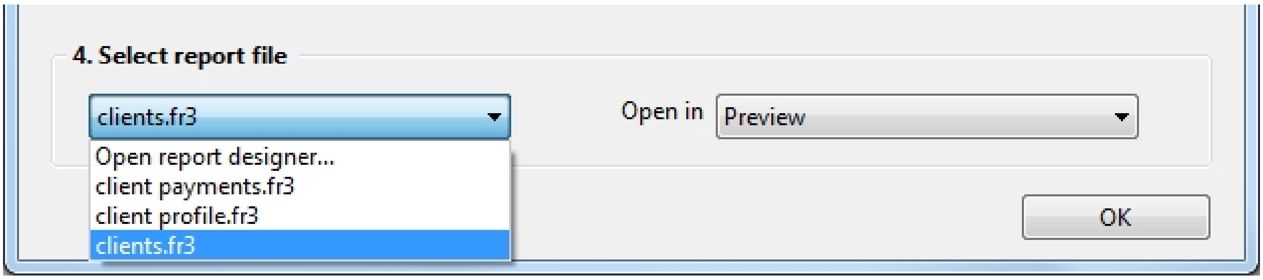


Name	Date modified	Type	Size
client payments.fr3	11/3/2020 2:54 PM	FR3 File	79 KB
client profile.fr3	11/3/2020 12:26 PM	FR3 File	76 KB

File name: clients.fr3

Save as type: Report (*.fr3)

It remains to go back to the settings of this button and select this report template to use for printing:



4. Select report file

clients.fr3

Open in Preview

Here we go. Now you can start the project again and click on this button, after which you will see a print-ready report with a list of clients.

Documentation on working with the report designer can be found here https://www.fast-report.com/public_download/UserManual-en.pdf

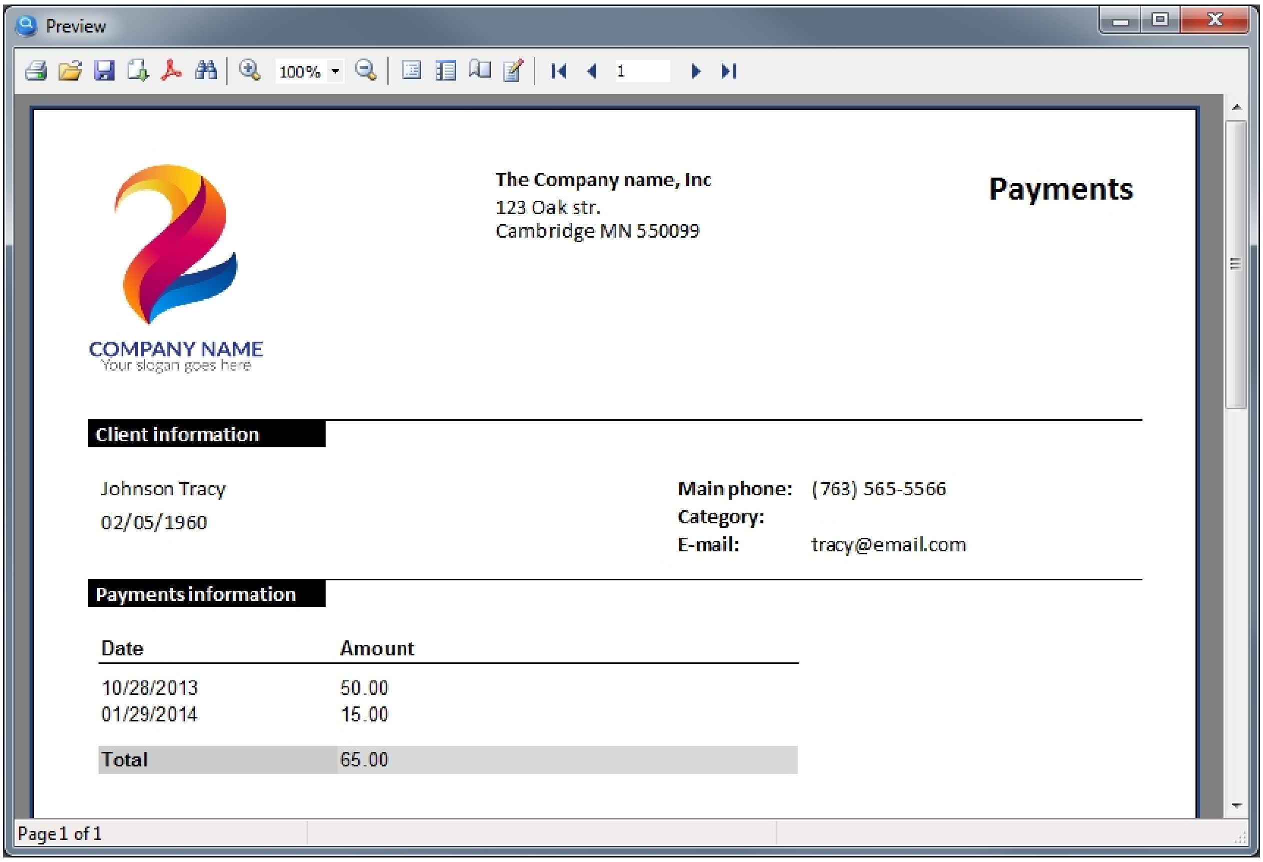
How to print a master-detail report

Description

Allows printing data from a database.

The setting of the Report button is almost the same as the setting of the Search button, since in fact both actions search the database and then display the search result, only in our case we will see the search result when printing. After setting up the button, you need to call the report designer to create a template, which will determine exactly how the document will look when printed, but everything in order.

As an example, we will implement printing of payments from the client. It will look as shown in the picture below:



The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

Client information

Johnson Tracy
02/05/1960

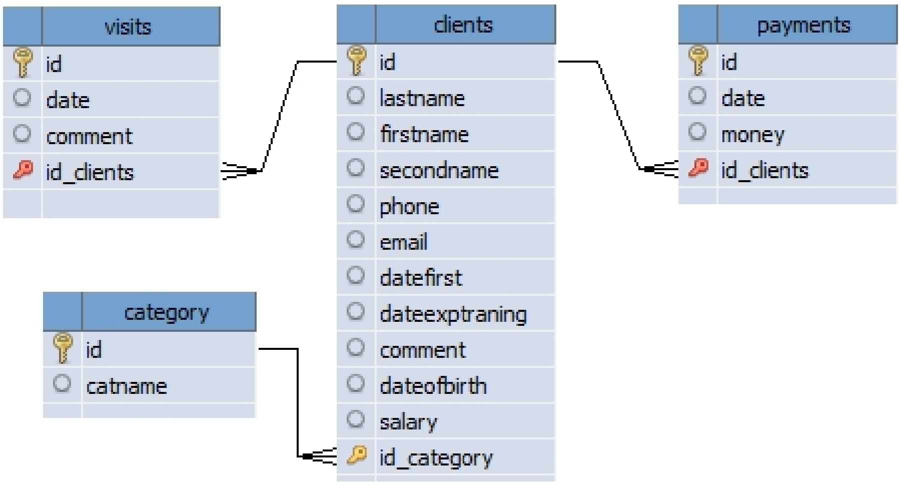
Main phone: (763) 565-5566
Category:
E-mail: tracy@email.com

Payments information

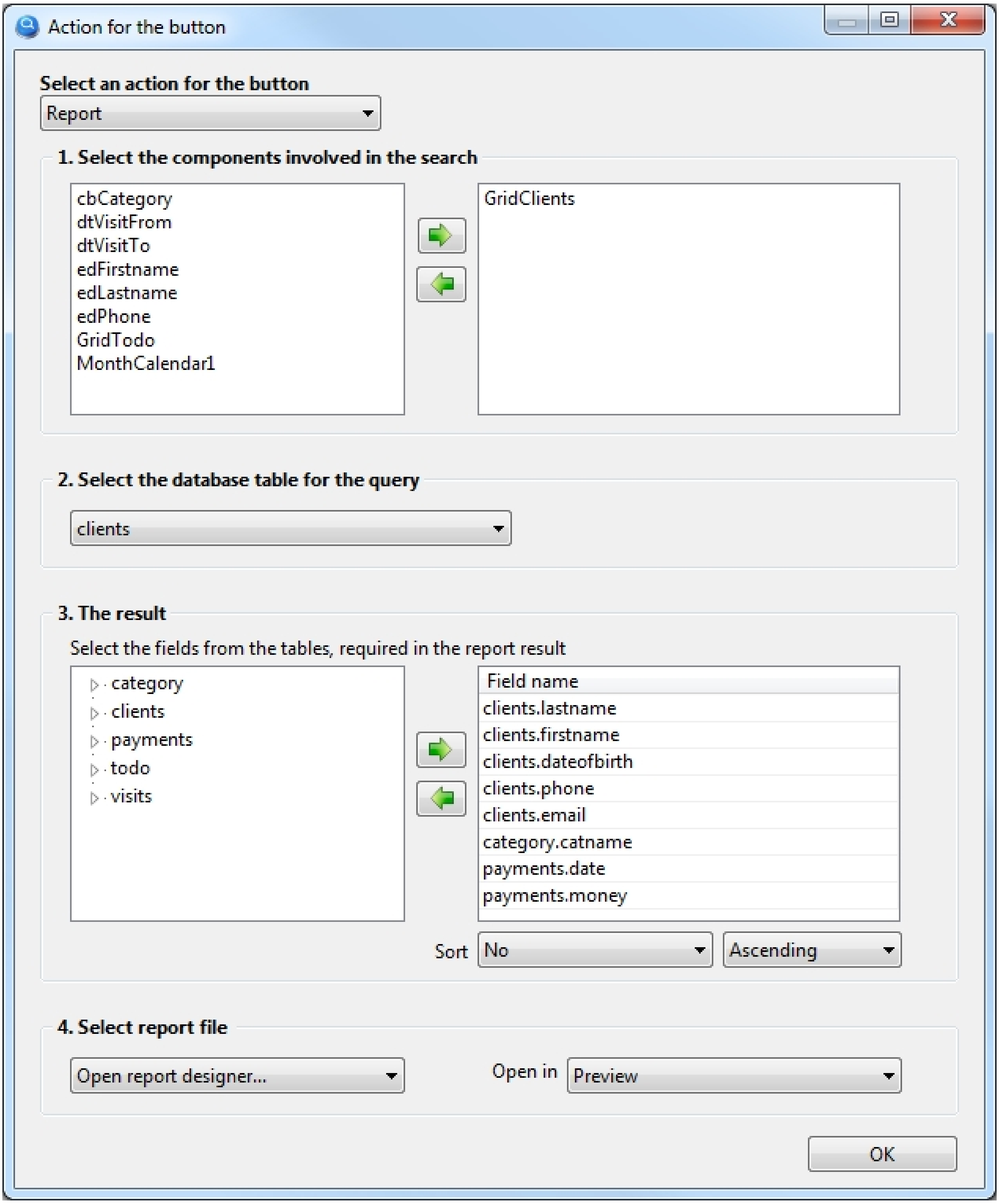
Date	Amount
10/28/2013	50.00
01/29/2014	15.00
Total	65.00

This type of reports is also called Master-Detail. In our example, the information about the client is Master (parent record), and the list of payments of this client is Detail (child records).

Database structure, which is used for this example:



Select the "Report" action for this button, the setting of this button is shown in the figure below:



Select an action for the button: Report

1. Select the components involved in the search

cbCategory
dtVisitFrom
dtVisitTo
edFirstname
edLastname
edPhone
GridTodo
MonthCalendar1

GridClients

2. Select the database table for the query: clients

3. The result

Select the fields from the tables, required in the report result

- category
- clients
- payments
- todo
- visits

Field name
clients.lastname
clients.firstname
clients.dateofbirth
clients.phone
clients.email
category.catname
payments.date
payments.money

Sort: No, Ascending

4. Select report file: Open report designer...

Open in: Preview

Let's consider this setting in detail by steps:

1. **Select the components involved in the search**

The GridClients component is selected here, where we see all our clients, so only the client that was selected in the GridClients component will be in the report. If there is no component in this list, then the report will receive all data from the selected database table.

2. **Select the database table for the query**

We will print information about the client, respectively select the "clients" database table.

3. **The result**

Select the table fields that we need in the report.

4. **Select report file**

Since the report template has yet to be created, this setting should select "Open report designer...".
A report template is a file that defines exactly what the report will look like, this file will be created in the report designer and will be saved in the Report folder of your project.

A nuance worth noting here. As you can see from the settings in p.2, to generate the result we select fields from 3 different tables (clients, category, payments), as a rule, in other programs you need to write an SQL query, which manually specifies how one table, links to another table and in what sequence.

The program My Visual Database tries to understand by itself how tables should be linked to each other in order to get exactly the data you expect, thus saving you from the need to learn the SQL query language.

Unfortunately, it is not always possible for the program to predict how tables should be linked to get the data you would like to see. Such a situation can occur when you need to link 3 or more tables and if there are no obvious links between them.

What should we do? On the "Database tables" tab, under each table, there is a checkbox "The table is a dictionary". On this tab, you should note which tables in your project are dictionaries. But how to understand which tables are dictionaries?

Examples of dictionary tables are the table containing country names, Statuses (Open, Closed), Types (Legal Entity, Physical Entity), name with prices, etc.

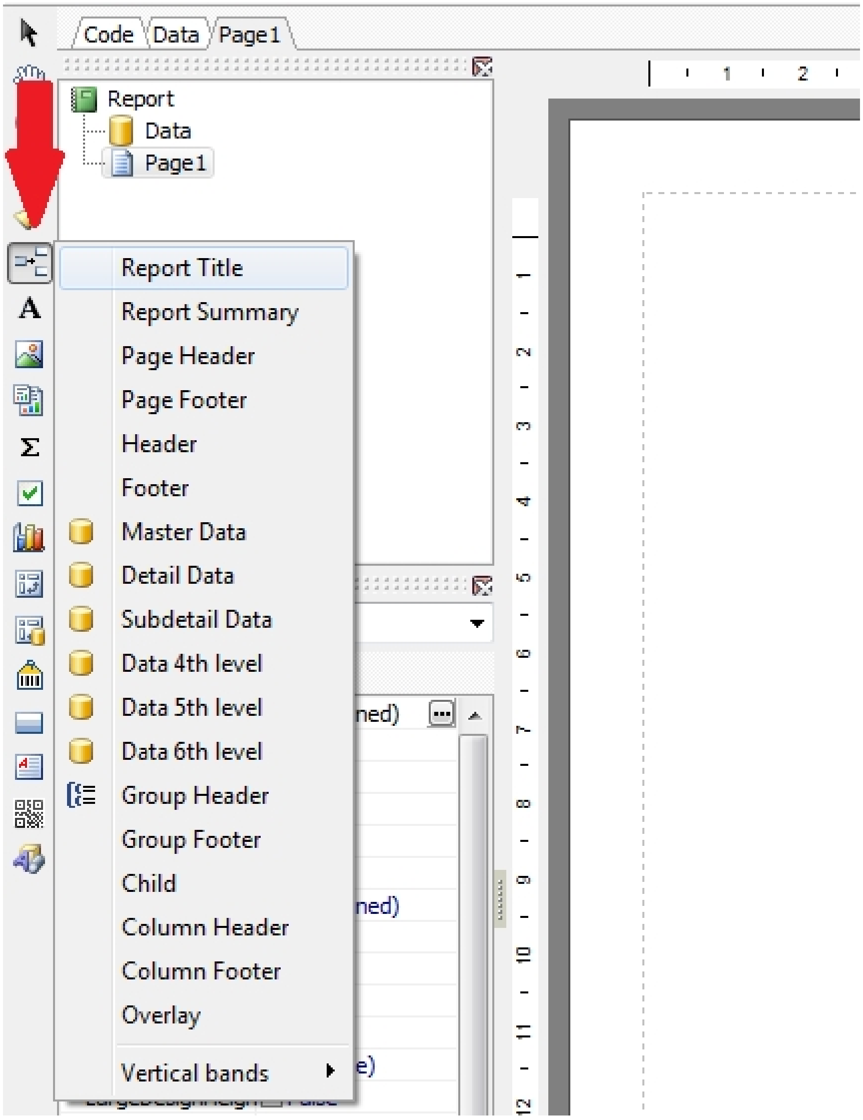
That is, such tables, which, as a rule, are filled in first at the start of work with the database and are not edited or edited rarely in the future.

In this case, the dictionary table can be referred to the "category" (client category), for this table you need to check the "Table is a dictionary" box under these tables, so you will help the program to link the tables correctly.

To create a report template, you need to run the project ▶
In the launched project, click this button with the report action, so you will open the report designer.

This report will be a bit more complicated than the previous one, as it will use so-called blocks (or Band). Blocks allow you to create almost any report structure. Some of them we will now get acquainted with.

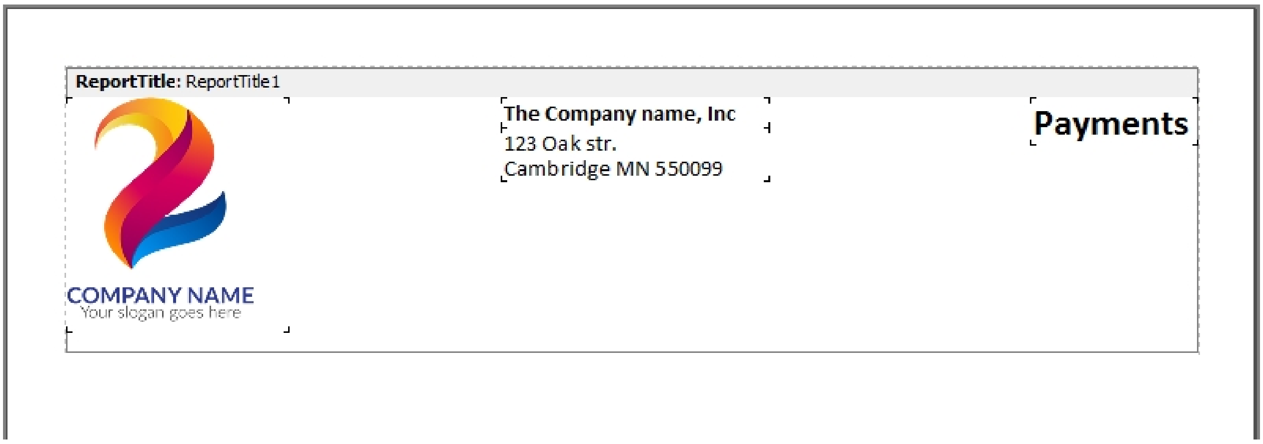
To view all available blocks, click on the left icon , then you will see a menu as shown in the figure below



Let's start creating a report using blocks and find out what they are for.

From the menu shown in the figure above, select the **Report Title** block and this block will appear in the report. The information in this block will be printed only on the first page of your report.

Place the logo and company information in this block. As a result, you should succeed as shown in the figure below:



ReportTitle: ReportTitle1

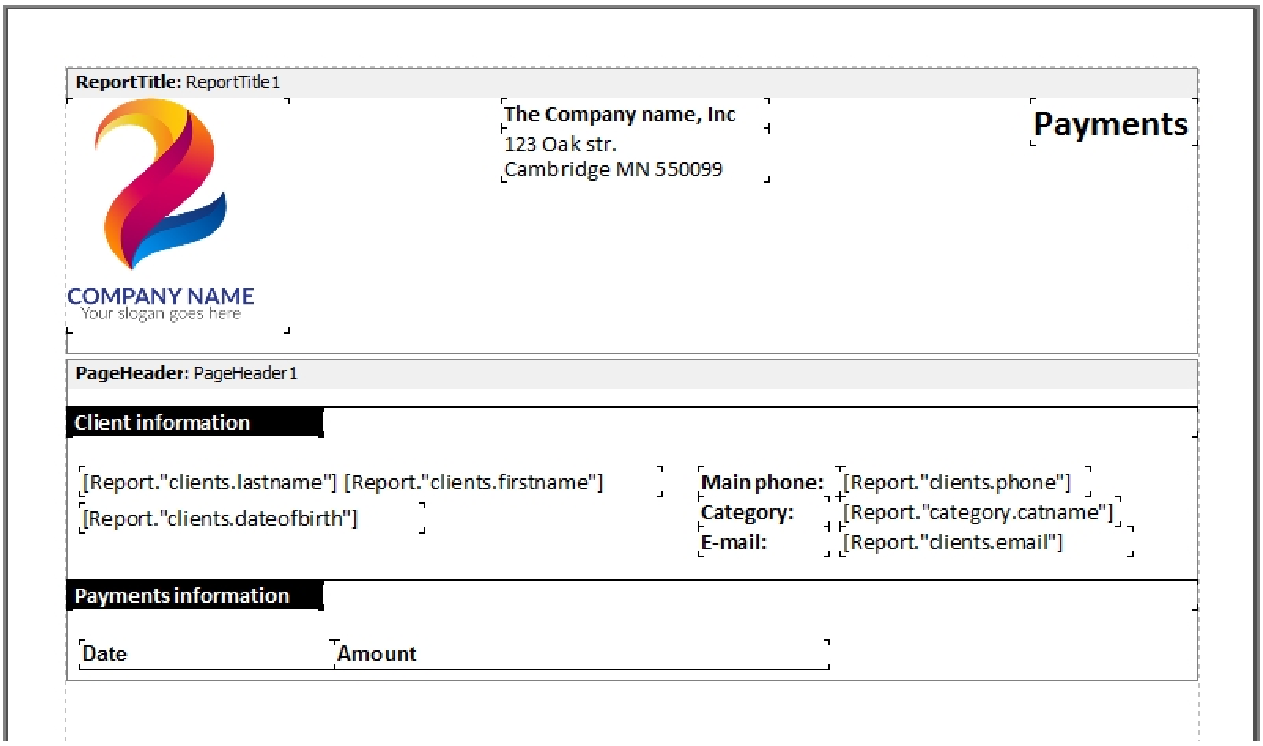
The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

You can skip this block if you do not need a title for your report.

Place the next **Page Header** block. The information placed in this block will be displayed on each printed page (in case your report does not fit on one page).

In this block we will place the information about the client. We will also place the headers for the table which will be located in the next block.
Place the text and data fields in this block as shown in the picture below:



ReportTitle: ReportTitle1

The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

PageHeader: PageHeader1

Client information

[Report."clients.lastname"] [Report."clients.firstname"]
[Report."clients.dateofbirth"]

Main phone: [Report."clients.phone"]
Category: [Report."category.catname"]
E-mail: [Report."clients.email"]

Payments information

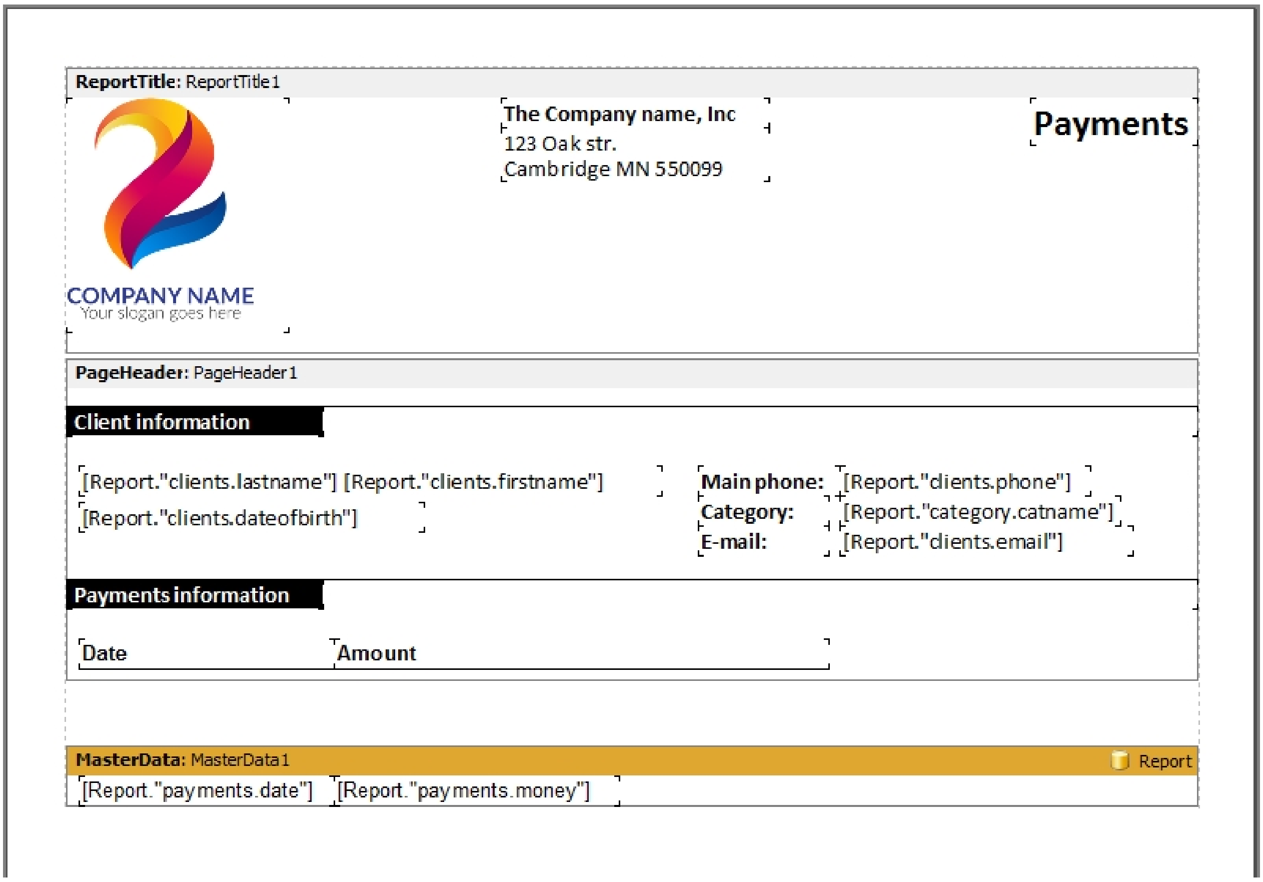
Date Amount

The next block to be placed in the report is **Master Data**.

This block is designed to output information in the form of a table. With the help of this block we will get a table where we will see all payments from this client.

Place this block in the report by selecting it from the menu. Before it appears in the report, you will see a window with the title **Select DataSet**, in which you need to select the data source. Select a data source from the list with the title **Report** and click **OK**.

In this block, place the database fields that you need in the table. As a result, you will see, as shown in the figure below:



ReportTitle: ReportTitle1

COMPANY NAME
Your slogan goes here

The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

PageHeader: PageHeader1

Client information

[Report."clients.lastname"] [Report."clients.firstname"]
[Report."clients.dateofbirth"]

Main phone: [Report."clients.phone"]
Category: [Report."category.catname"]
E-mail: [Report."clients.email"]

Payments information

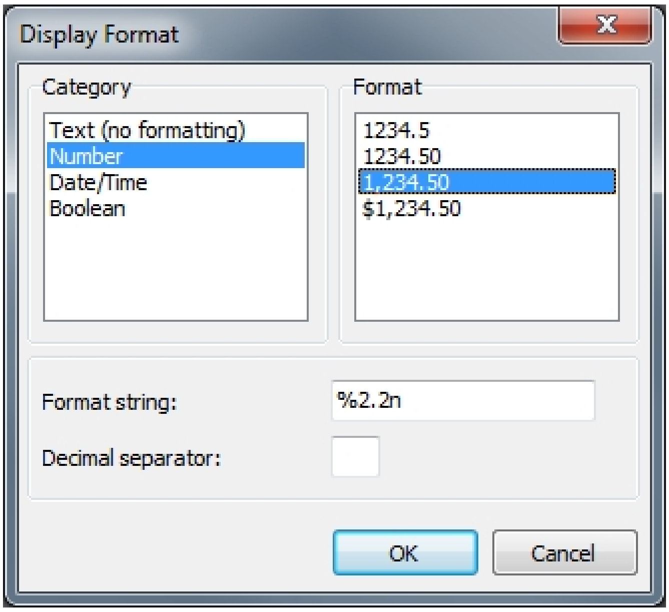
Date Amount

MasterData: MasterData1 Report

[Report."payments.date"] [Report."payments.money"]

Pay attention to the [Report. "payments.money"] field, which displays the price of the payment. Let's apply monetary formatting to this field, i.e. add a separator of thousands and mandatory two decimal places. To do this, right-click on this field and select "Display Format..." in the menu.

Choose the value Number in the Category list and 1,234.50 in the Format list, then click OK.



Display Format

Category: Text (no formatting), **Number** (selected), Date/Time, Boolean

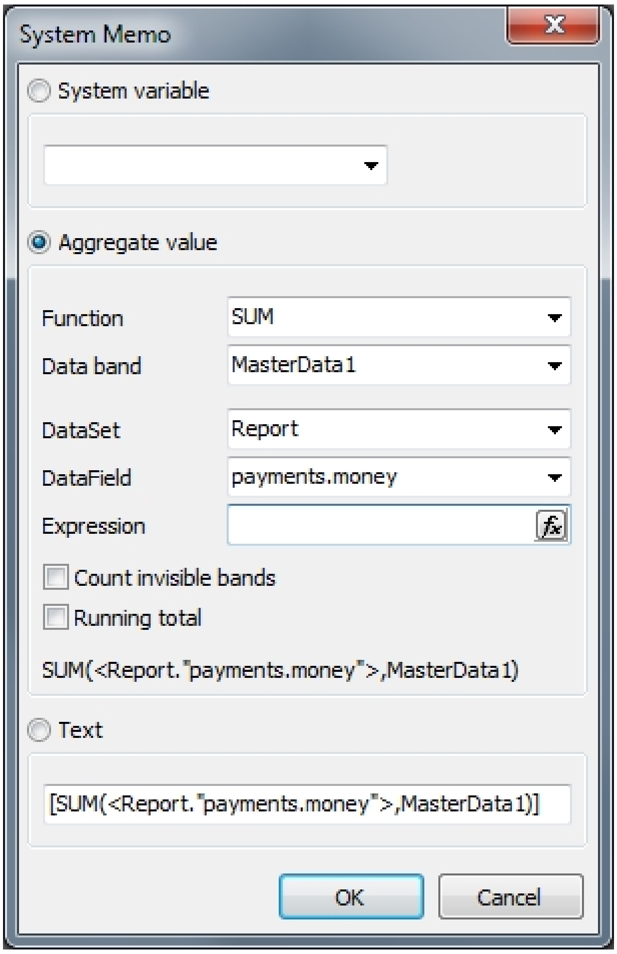
Format: 1234.5, 1234.50, **1,234.50** (selected), $1,234.50

Format string: %2.2n

Decimal separator:

Place the next **Footer** block. This block will print once at the end of the report, i.e. you will not see it on every page of your report.
In this block you can calculate, for example, the total amount of all payments from the client.

To calculate the total amount, place the **System text** Σ component in this block. Once you have placed this component, a dialog box will appear and you need to configure it as shown in the figure below:



System Memo

○ System variable

● Aggregate value

Function	SUM
Data band	MasterData1
DataSet	Report
DataField	payments.money
Expression	

□ Count invisible bands

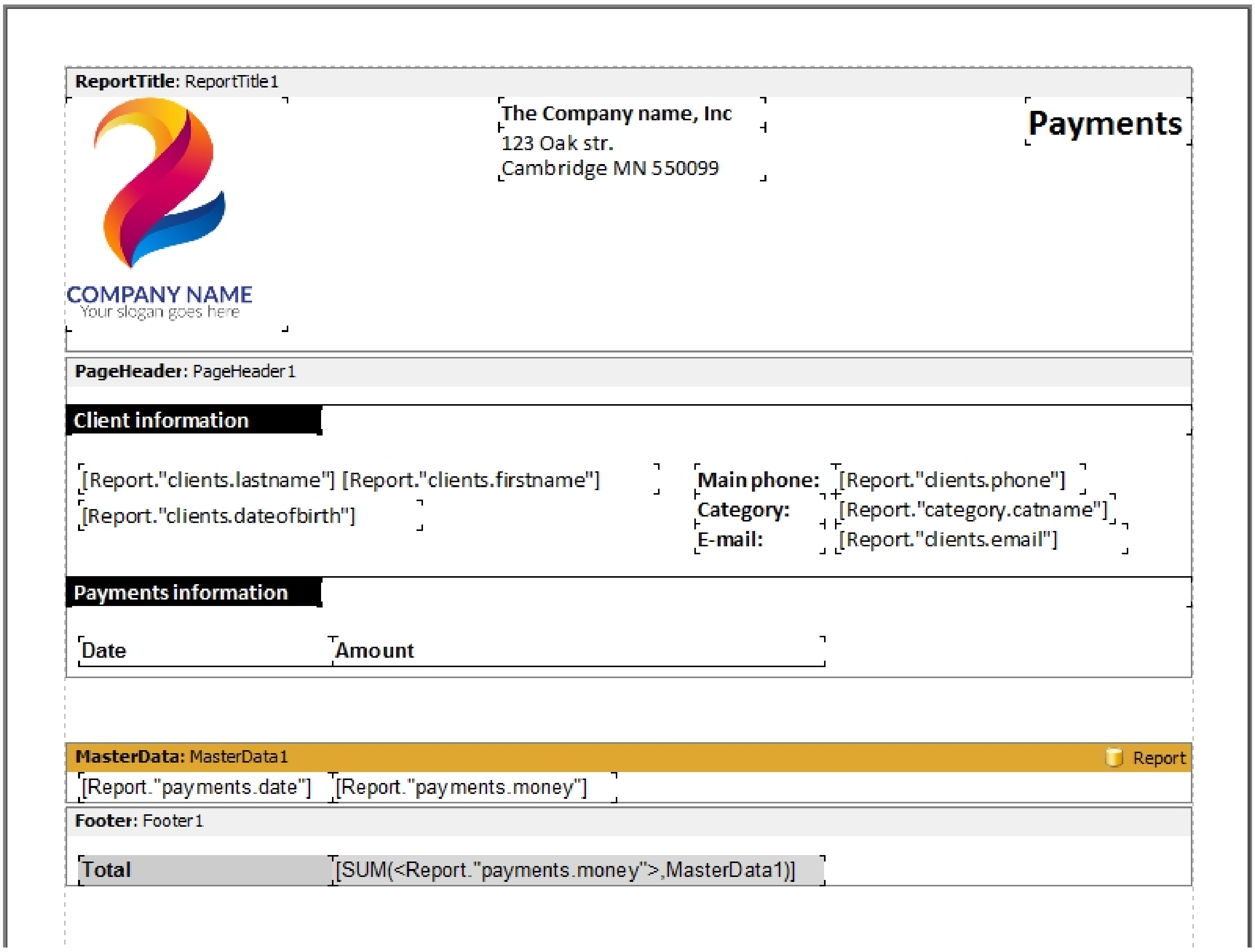
□ Running total

SUM(<Report."payments.money">,MasterData1)

○ Text

[SUM(<Report."payments.money">,MasterData1)]

After that, your report template should look as shown in the figure below:



ReportTitle: ReportTitle1

COMPANY NAME
Your slogan goes here

The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

PageHeader: PageHeader1

Client information

[Report."clients.lastname"] [Report."clients.firstname"]
[Report."clients.dateofbirth"]

Main phone: [Report."clients.phone"]
Category: [Report."category.catname"]
E-mail: [Report."clients.email"]

Payments information

Date Amount

MasterData: MasterData1 Report

[Report."payments.date"] [Report."payments.money"]

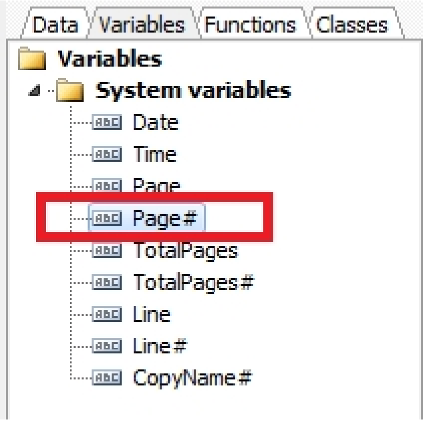
Footer: Footer1

Total [SUM(<Report."payments.money">,MasterData1)]

You can also apply monetary formatting to this field, as described in the previous step.

And the last block that we will place in the report is **Page Footer**, this block will be printed on each page of the report, we use it to numerate the pages.

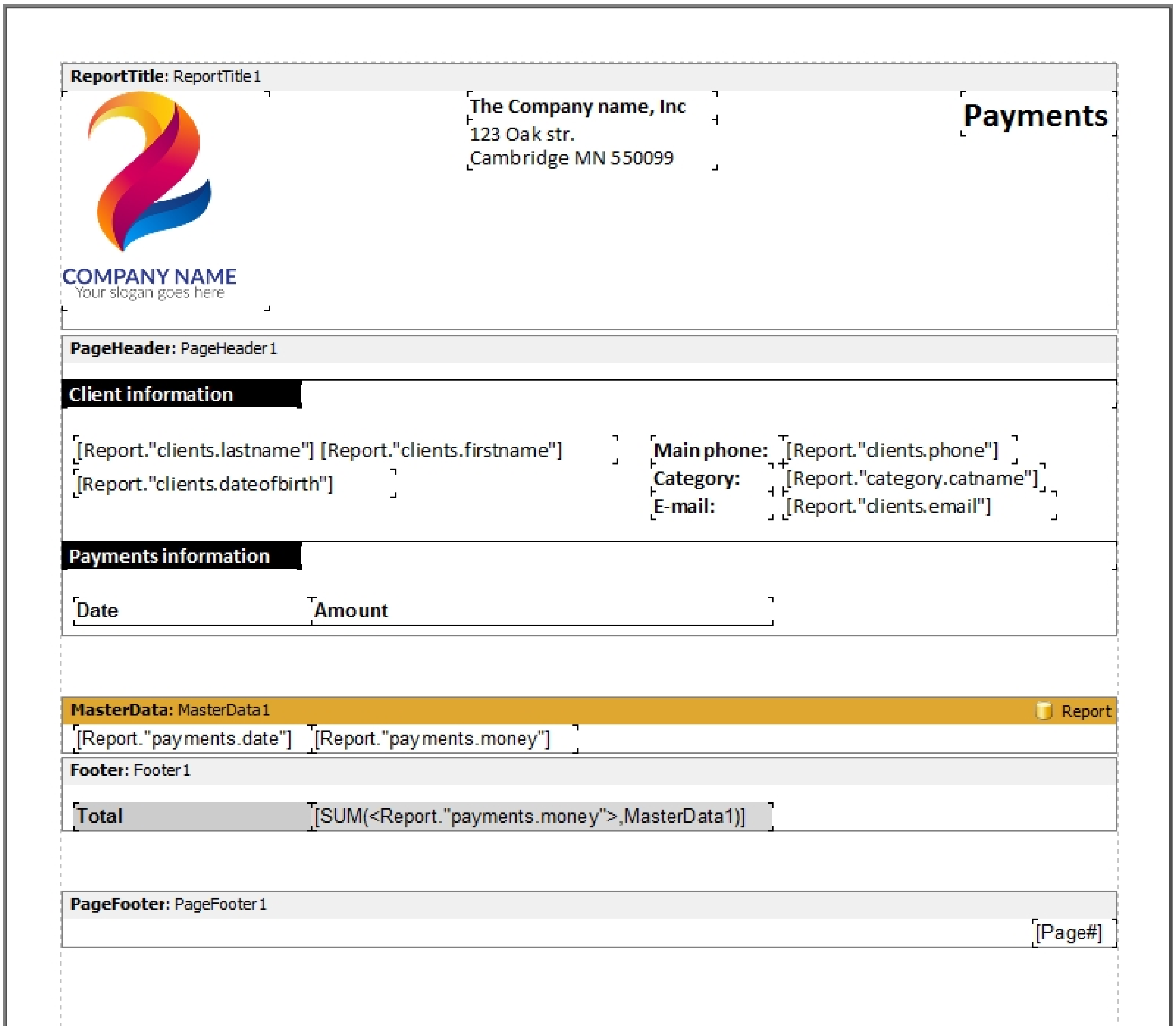
Place the **Page#** system variable in this block from the right side of the report designer (Variables tab), just drag and drop:



Data | Variables | Functions | Classes

- Variables
  - System variables
    - Date
    - Time
    - Page
    - Page#
    - TotalPages
    - TotalPages#
    - Line
    - Line#
    - CopyName#

Now your report template is fully ready



ReportTitle: ReportTitle1

COMPANY NAME
Your slogan goes here

The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

PageHeader: PageHeader1

Client information

[Report."clients.lastname"] [Report."clients.firstname"]
[Report."clients.dateofbirth"]

Main phone: [Report."clients.phone"]
Category: [Report."category.catname"]
E-mail: [Report."clients.email"]

Payments information

Date **Amount**

MasterData: MasterData1 — Report

[Report."payments.date"] [Report."payments.money"]

Footer: Footer1

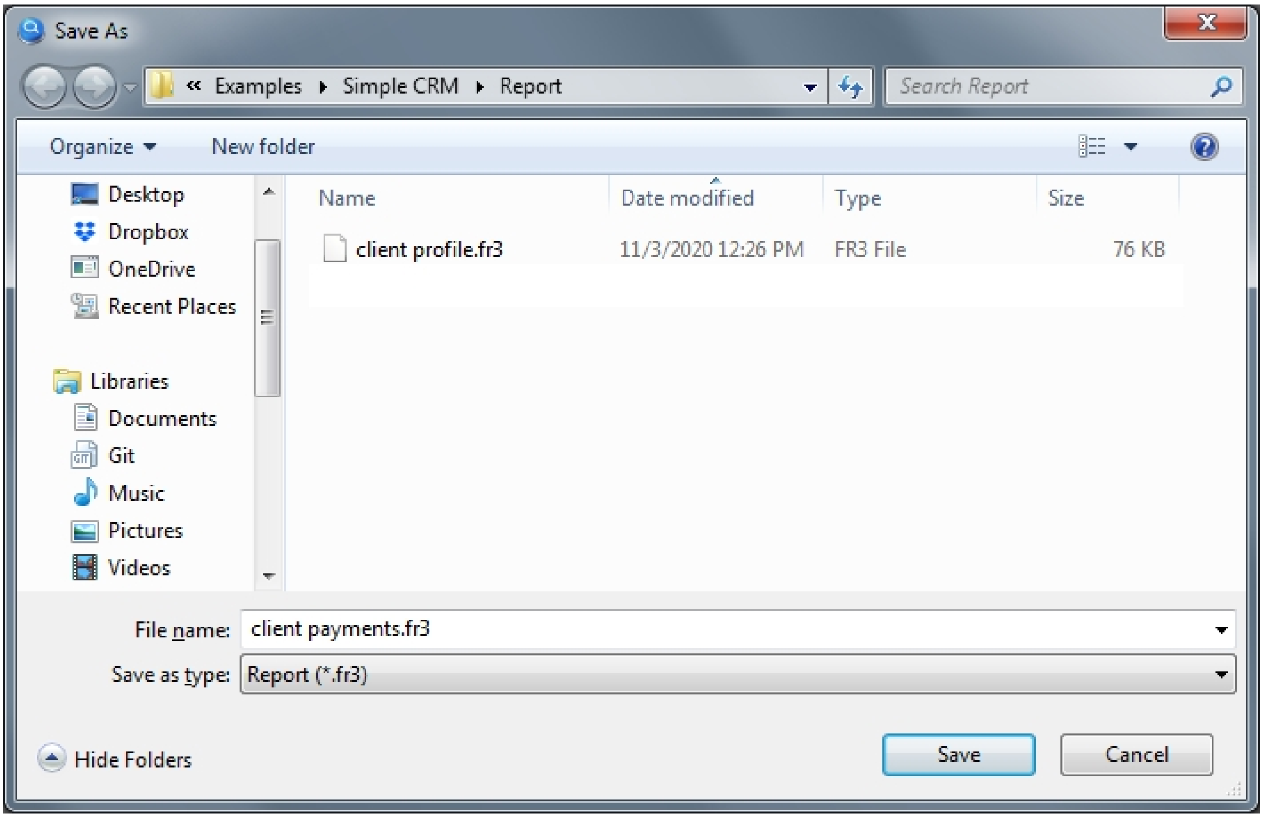
Total [SUM(<Report."payments.money">,MasterData1)]

PageFooter: PageFooter1

[Page#]

Go to the File > Preview menu or just press Ctrl+P to see what your report will look like.

Save the report template to the **Report** folder of your project, menu **File > Save As**... Give the file a name such as "**client payments**" as shown in the figure below:

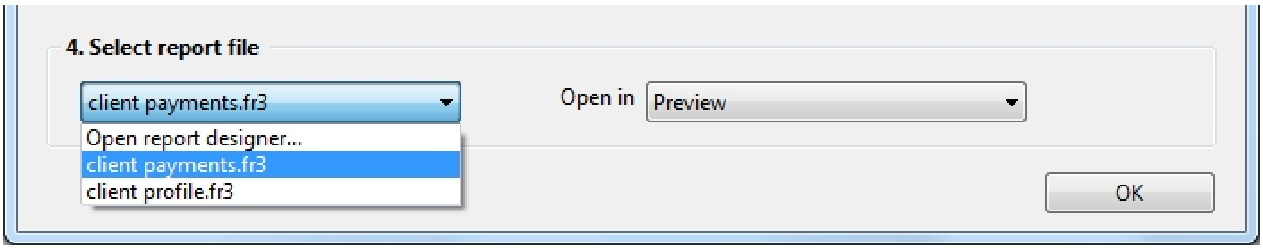


Name	Date modified	Type	Size
client profile.fr3	11/3/2020 12:26 PM	FR3 File	76 KB

File name: client payments.fr3

Save as type: Report (*.fr3)

It remains to go back to the settings of this button and select this report template to use for printing:



4. Select report file

client payments.fr3

Open in: Preview

Here we go. Now you can start the project again. After selecting the desired client in the GridClients component, click on this button and you'll see a printable report with the selected client data and its payment list.

Sometimes it is convenient to place this button for printing directly on the form of adding/editing a record. In this case, leave the list blank in the button settings "**1. Select the components involved in the search**". If the button with the "Report" action is placed on the form intended for adding/editing a record, it automatically recognizes which one should be sent to print.

Documentation on working with the report designer can be found here https://www.fast-report.com/public_download/UserManual-en.pdf

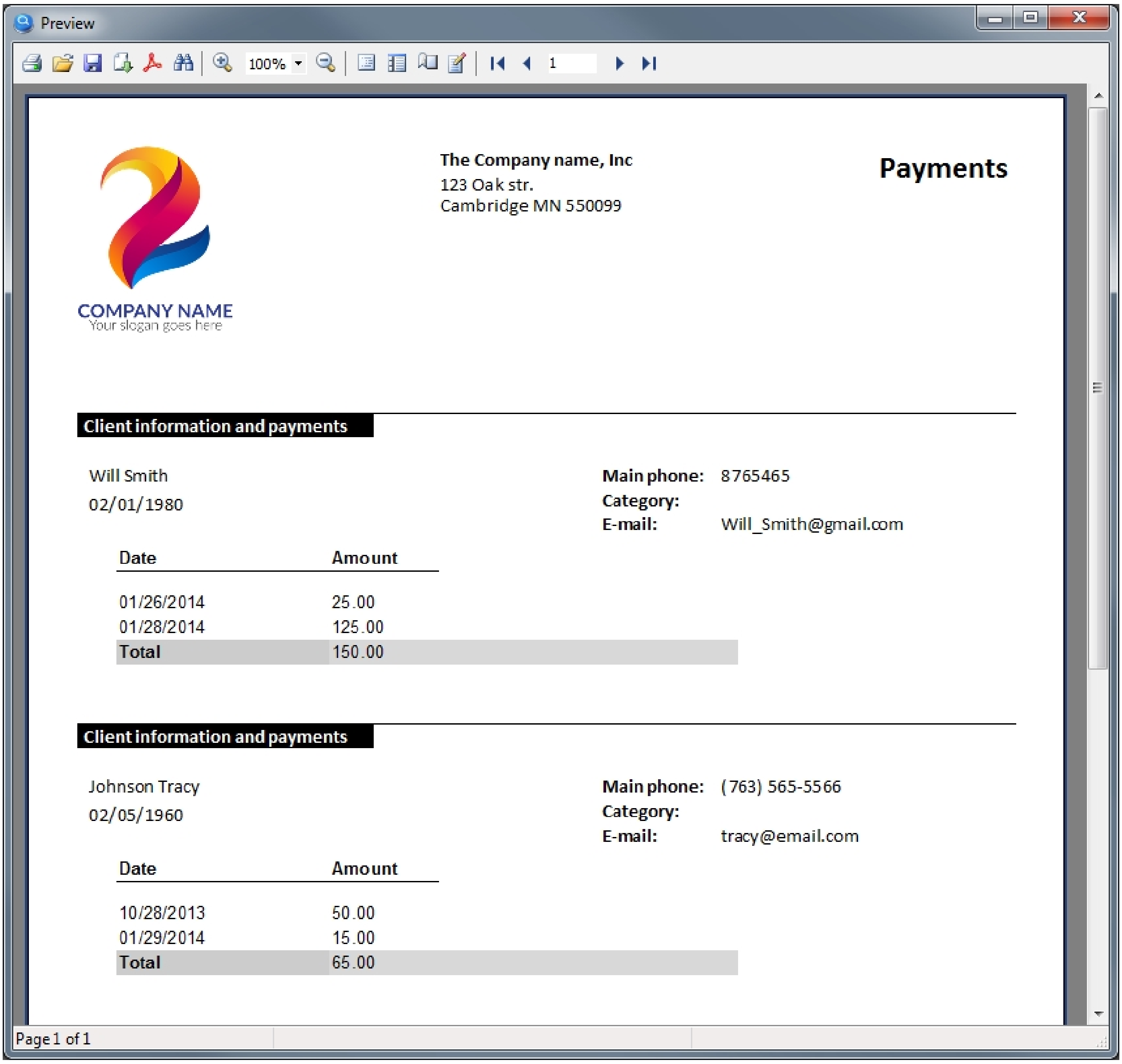
How to print a master-detail report with grouping

Description

Allows printing data from a database.

The setting of the Report button is almost the same as the setting of the Search button, since in fact both actions search the database and then display the search result, only in our case we will see the search result when printing. After setting up the button, you need to call the report designer to create a template, which will determine exactly how the document will look when printed, but everything in order.

As an example, we will implement printing of payments from the clients. It will look as shown in the picture below:



The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

Client information and payments

Will Smith
02/01/1980

Main phone: 8765465
Category:
E-mail: Will_Smith@gmail.com

Date	Amount
01/26/2014	25.00
01/28/2014	125.00
Total	150.00

Client information and payments

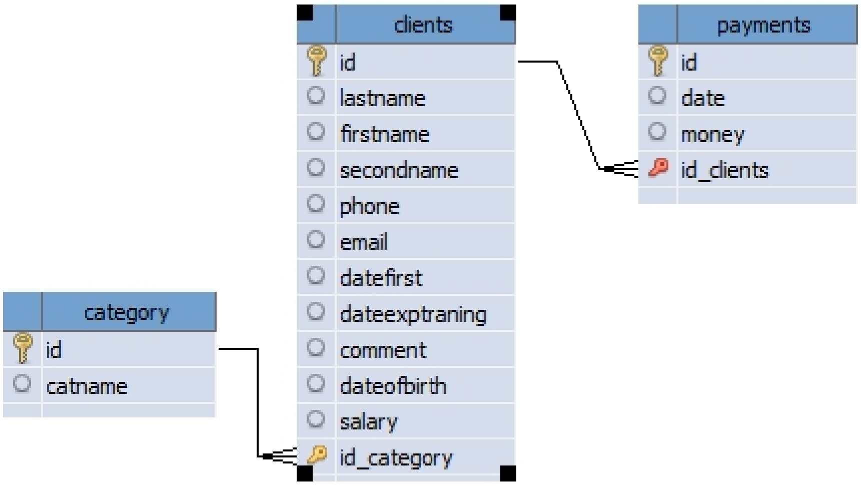
Johnson Tracy
02/05/1960

Main phone: (763) 565-5566
Category:
E-mail: tracy@email.com

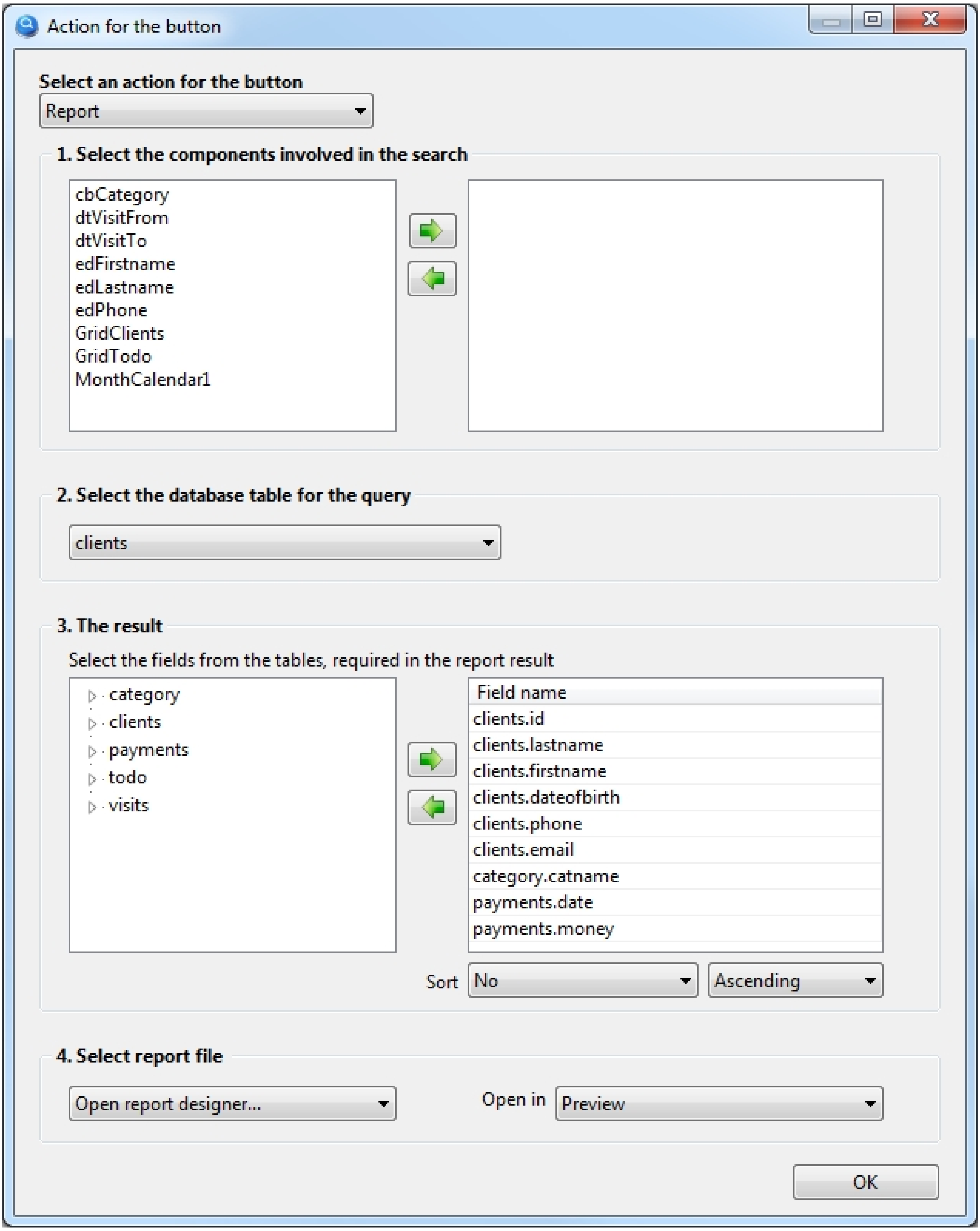
Date	Amount
10/28/2013	50.00
01/29/2014	15.00
Total	65.00

This type of reports is also called Master-Detail with grouping. In our example, the client information is Master (parent record), and the list of payments from these clients is Detail (child records), which are grouped by client.

Database structure, which is used for this example:



Select the "Report" action for this button, the setting of this button is shown in the figure below:



Select an action for the button: Report

1. Select the components involved in the search

cbCategory
dtVisitFrom
dtVisitTo
edFirstname
edLastname
edPhone
GridClients
GridTodo
MonthCalendar1

2. Select the database table for the query: clients

3. The result

Select the fields from the tables, required in the report result

category
clients
payments
todo
visits

Field name
clients.id
clients.lastname
clients.firstname
clients.dateofbirth
clients.phone
clients.email
category.catname
payments.date
payments.money

Sort: No, Ascending

4. Select report file: Open report designer...

Open in: Preview

Let's consider this setting in detail by steps:

1. **Select the components involved in the search**
This list is empty, so the report will contain data on all clients that are present in the database. If necessary, you can add components that will only include clients that meet your criteria in the report.

2. **Select the database table for the query**

We will print information about the client, respectively select the "clients" database table.

3. **The result**

Select the table fields that we need in the report.

4. **Select report file**

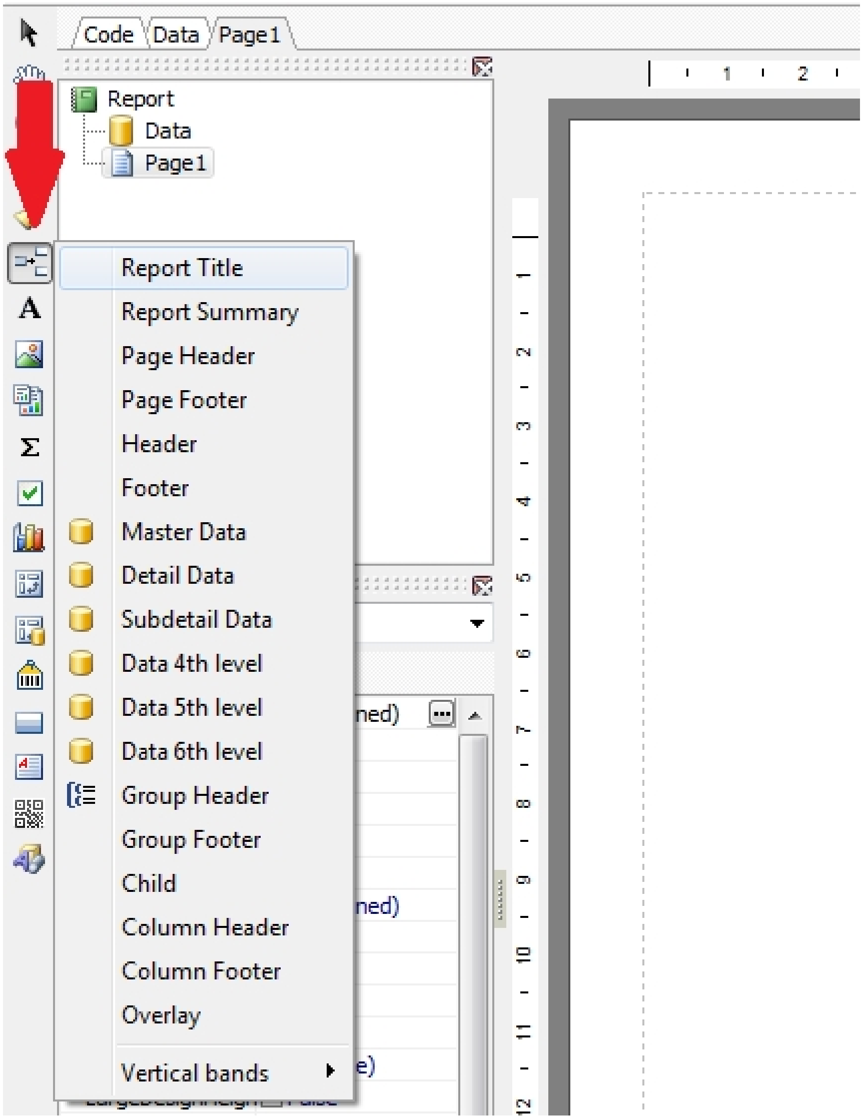
Since the report template has yet to be created, this setting should select "Open report designer...".
A report template is a file that defines exactly what the report will look like, this file will be created in the report designer and will be saved in the Report folder of your project.

> A nuance worth noting here. As you can see from the settings in p.2, to generate the result we select fields from 3 different tables (clients, category, payments), as a rule, in other programs you need to write an SQL query, which manually specifies how one table, links to another table and in what sequence.
>
> The program My Visual Database tries to understand by itself how tables should be linked to each other in order to get exactly the data you expect, thus saving you from the need to learn the SQL query language.
>
> Unfortunately, it is not always possible for the program to predict how tables should be linked to get the data you would like to see. Such a situation can occur when you need to link 3 or more tables and if there are no obvious links between them.
>
> What should we do? On the "Database tables" tab, under each table, there is a checkbox "The table is a dictionary". On this tab, you should note which tables in your project are dictionaries. But how to understand which tables are dictionaries?
>
> Examples of dictionary tables are the table containing country names, Statuses (Open, Closed), Types (Legal Entity, Physical Entity), name with prices, etc.
>
> That is, such tables, which, as a rule, are filled in first at the start of work with the database and are not edited or edited rarely in the future.
>
> In this case, the dictionary table can be referred to the "category" (client category), for this table you need to check the "Table is a dictionary" box under these tables, so you will help the program to link the tables correctly.

To create a report template, you need to run the project ▶
In the launched project, click this button with the report action, so you will open the report designer.

This report will be a bit more complicated than the previous one, as it will use so-called blocks (or Band). Blocks allow you to create almost any report structure. Some of them we will now get acquainted with.

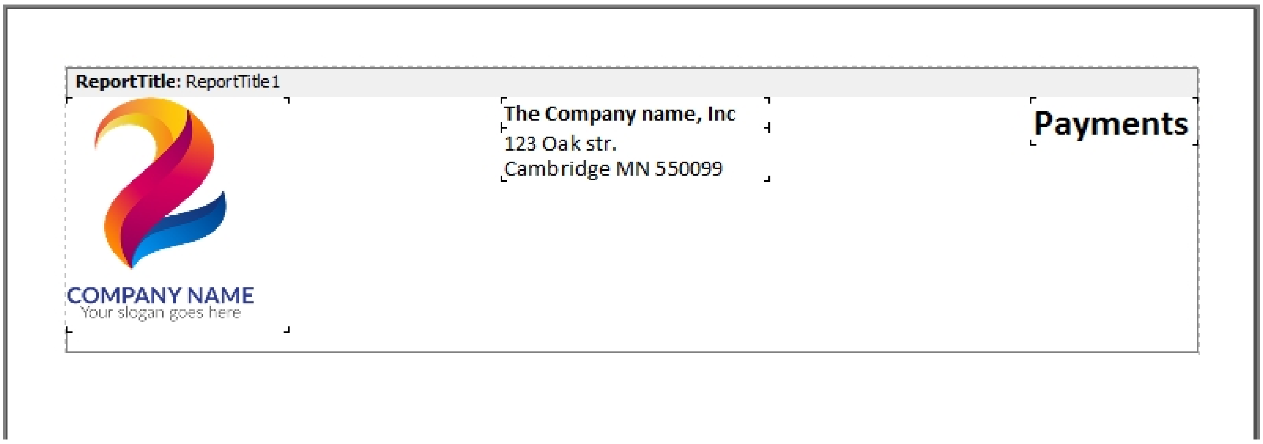
To view all available blocks, click on the left icon , then you will see a menu as shown in the figure below



Let's start creating a report using blocks and find out what they are for.

From the menu shown in the figure above, select the **Report Title** block and this block will appear in the report. The information in this block will be printed only on the first page of your report.

Place the logo and company information in this block. As a result, you should succeed as shown in the figure below:



ReportTitle: ReportTitle1

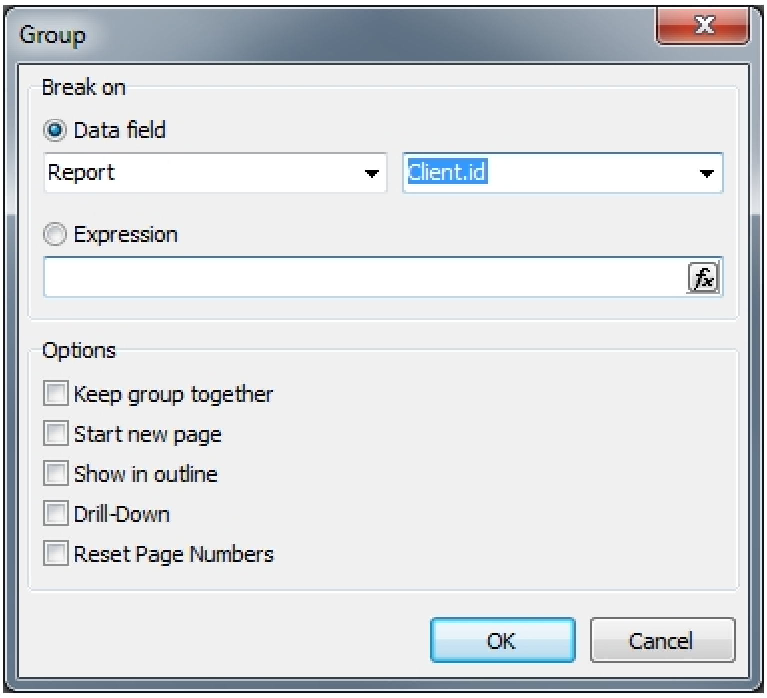
The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

You can skip this block if you do not need a title for your report.

Place the next **Group Header** block. This block allows grouping the data. In our case we will group data by client, so the MasterData block below will display only those data that belong to the current client.

When this block is placed, you need to specify by which field of the database table the data should be grouped, in our case we have added the **Client.id** field specifically for this purpose. This field unambiguously identifies the client, even if by mistake there are two clients with the same name in the database.



Group

Break on

● Data field

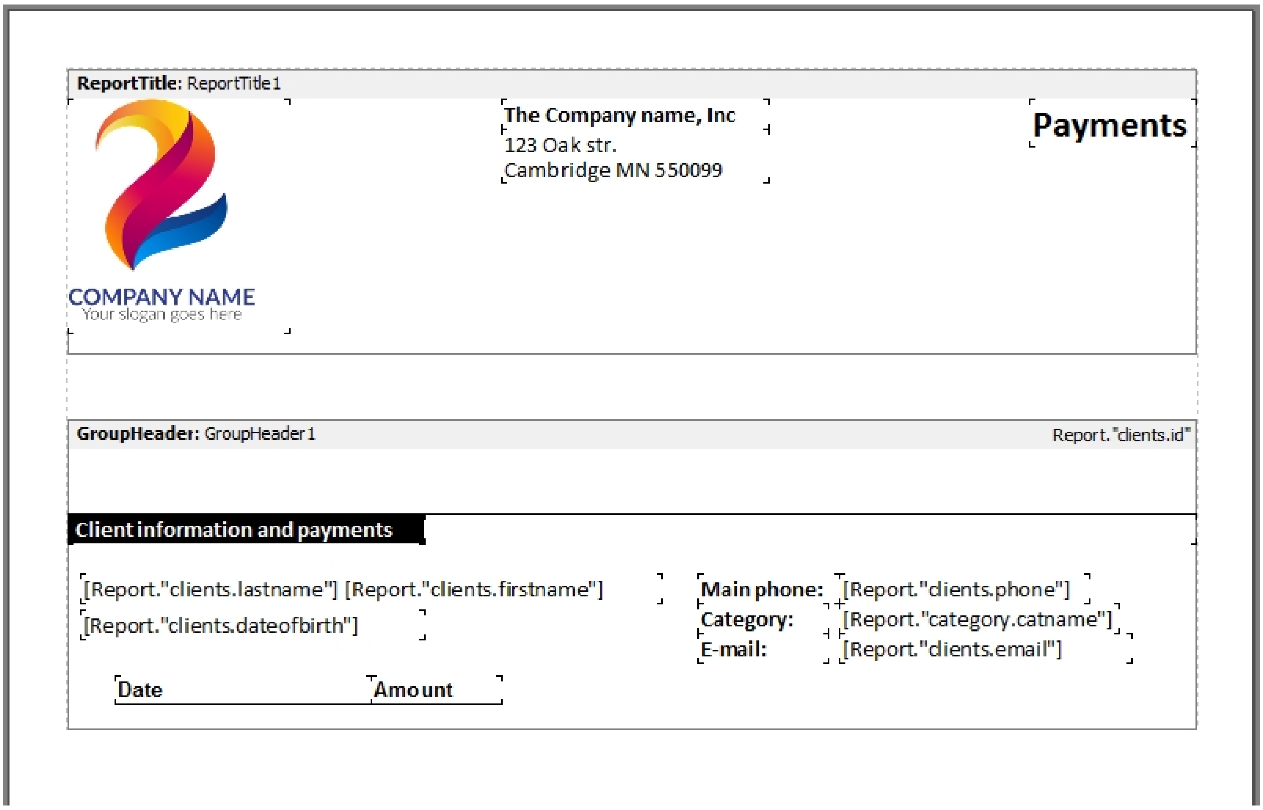
Report | Client.id

○ Expression

Options

□ Keep group together
□ Start new page
□ Show in outline
□ Drill-Down
□ Reset Page Numbers

In this block we will place the information about the client. We will also place the headers for the table that will be located in the next block.
Place the text and data fields in this block, as shown in the figure below:



ReportTitle: ReportTitle1

The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

GroupHeader: GroupHeader1 Report."clients.id"

Client information and payments

[Report."clients.lastname"] [Report."clients.firstname"]
[Report."clients.dateofbirth"]

Main phone: [Report."clients.phone"]
Category: [Report."category.catname"]
E-mail: [Report."clients.email"]

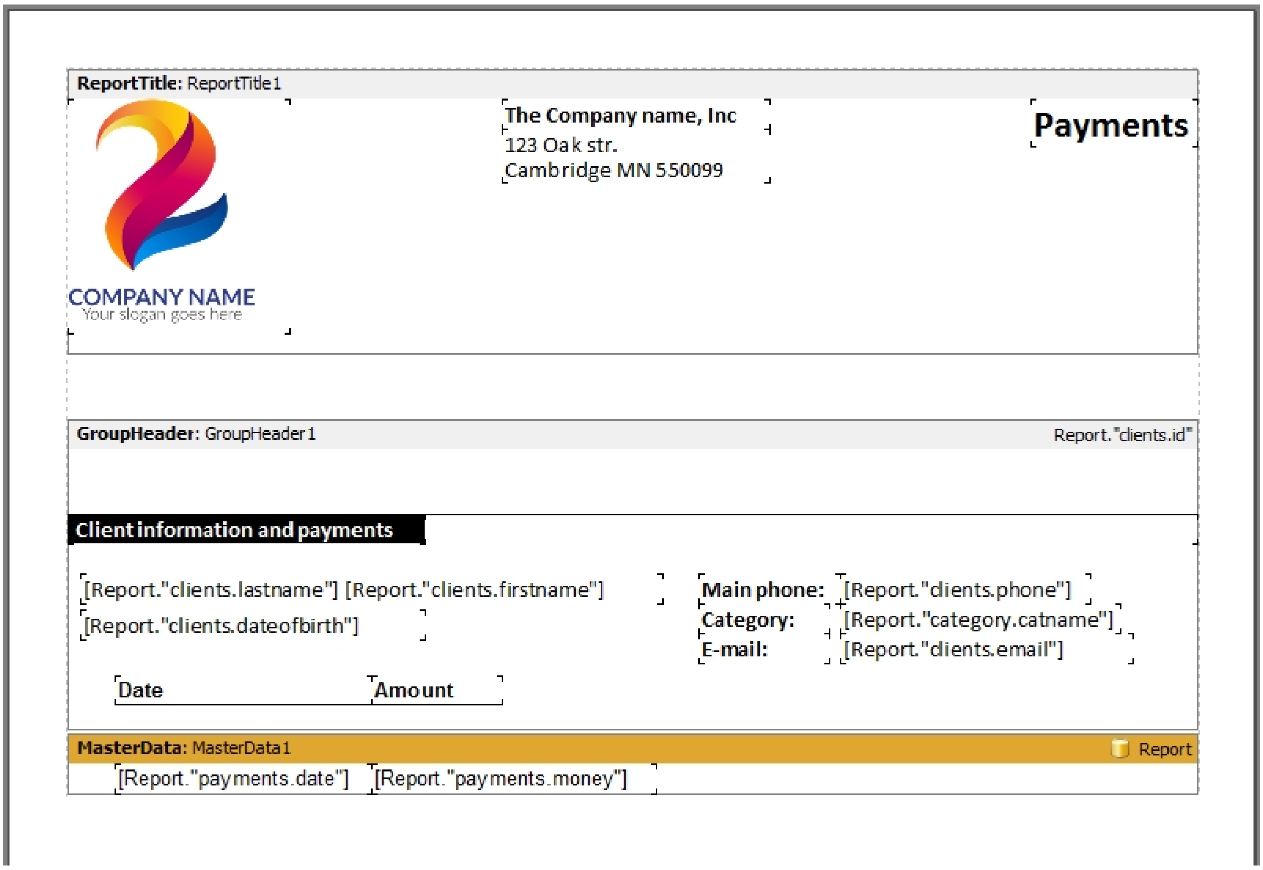
Date **Amount**

The next block to be placed in the report is **Master Data**. This block is designed to output information in the form of a table.

Due to the fact that in the previous step we have placed the GroupHeader block, in the Master Data block we will see data that belong to the current client, namely, data on the rented equipment.

Place this block in the report by selecting it from the menu. Before it appears in the report, you will see a window with the title **Select DataSet**, in which you need to select the data source. Select a data source with the **Report** title from the list and click **OK**.

In this block, place the database fields from which the table will be formed. As a result, you should succeed as shown in the figure below:



ReportTitle: ReportTitle1

The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

GroupHeader: GroupHeader1 Report."clients.id"

Client information and payments

[Report."clients.lastname"] [Report."clients.firstname"]
[Report."clients.dateofbirth"]

Main phone: [Report."clients.phone"]
Category: [Report."category.catname"]
E-mail: [Report."clients.email"]

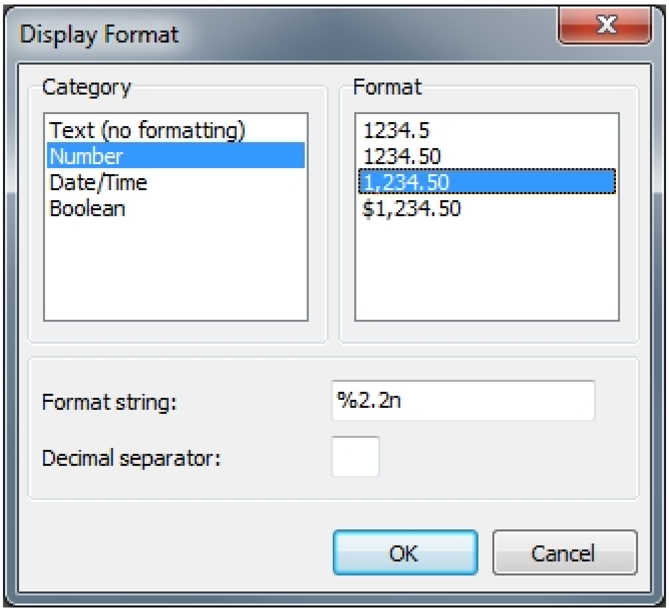
Date **Amount**

MasterData: MasterData1 Report

[Report."payments.date"] [Report."payments.money"]

Pay attention to the [Report. "payments.money"] field, which displays the price of the payment. Let's apply monetary formatting to this field, i.e. add a separator of thousands and mandatory two decimal places. To do this, right-click on this field and select "Display Format..." in the menu.

Choose the value Number in the Category list and 1,234.50 in the Format list, then click OK.



Display Format

Category: Text (no formatting), **Number** (selected), Date/Time, Boolean

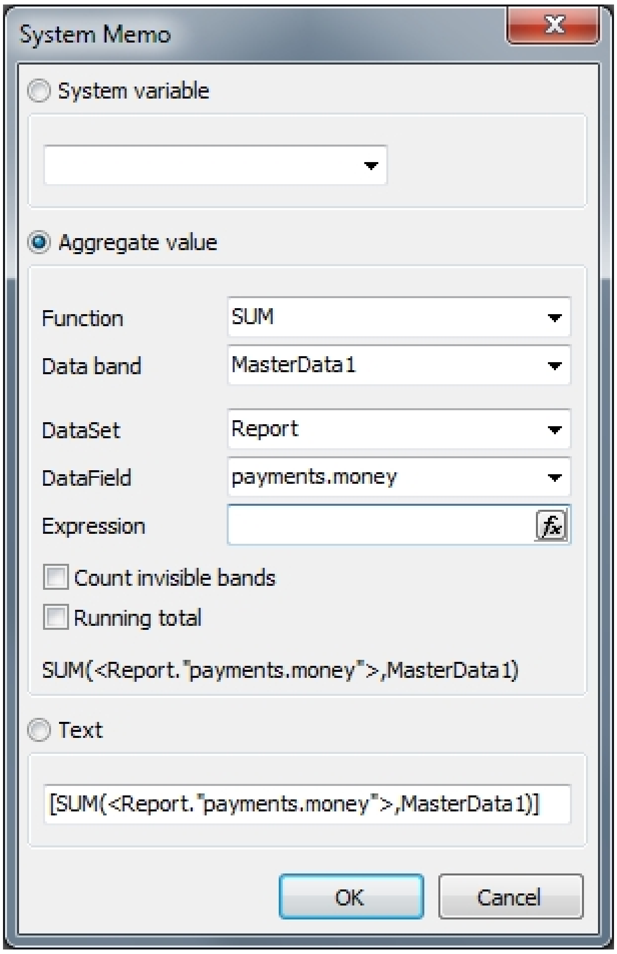
Format: 1234.5, 1234.50, **1,234.50** (selected), $1,234.50

Format string: %2.2n

Decimal separator:

Place the next **Group Footer** block. In this block you can calculate the total amount of payments from the current client.

To calculate the total amount, place the **System text** Σ component in this block. Once you have placed this component, a dialog box will appear and you need to configure it as shown in the figure below:



System Memo

○ System variable

● Aggregate value

Function	SUM
Data band	MasterData1
DataSet	Report
DataField	payments.money
Expression	

□ Count invisible bands

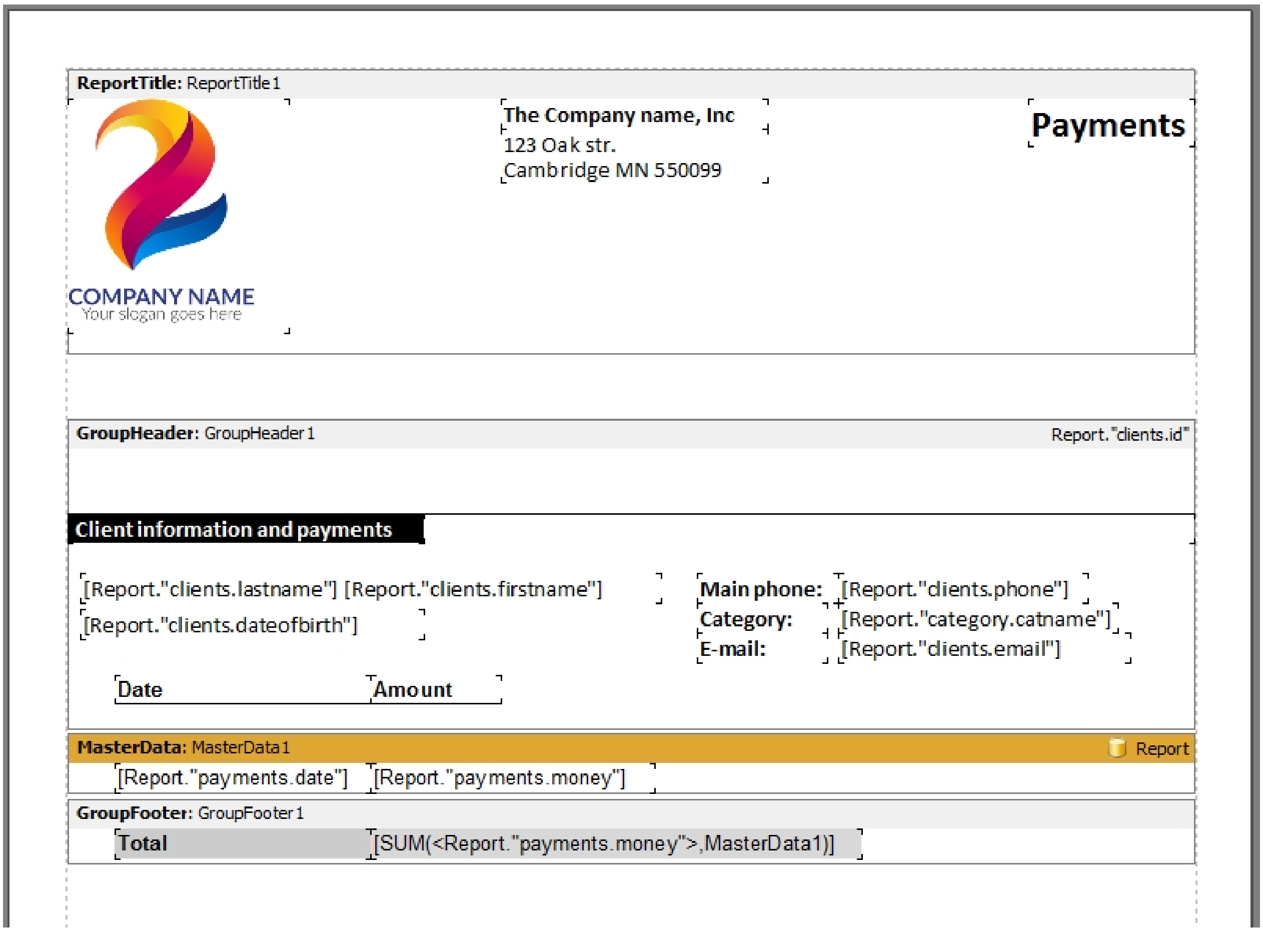
□ Running total

SUM(<Report."payments.money">,MasterData1)

○ Text

[SUM(<Report."payments.money">,MasterData1)]

After that, your report template should look as shown in the figure below:



ReportTitle: ReportTitle1

COMPANY NAME
Your slogan goes here

The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

GroupHeader: GroupHeader1 Report."clients.id"

Client information and payments

[Report."clients.lastname"] [Report."clients.firstname"]
[Report."clients.dateofbirth"]

Main phone: [Report."clients.phone"]
Category: [Report."category.catname"]
E-mail: [Report."clients.email"]

Date Amount

MasterData: MasterData1 Report

[Report."payments.date"] [Report."payments.money"]

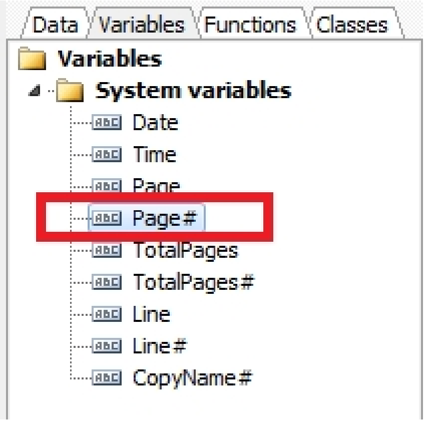
GroupFooter: GroupFooter1

Total [SUM(<Report."payments.money">,MasterData1)]

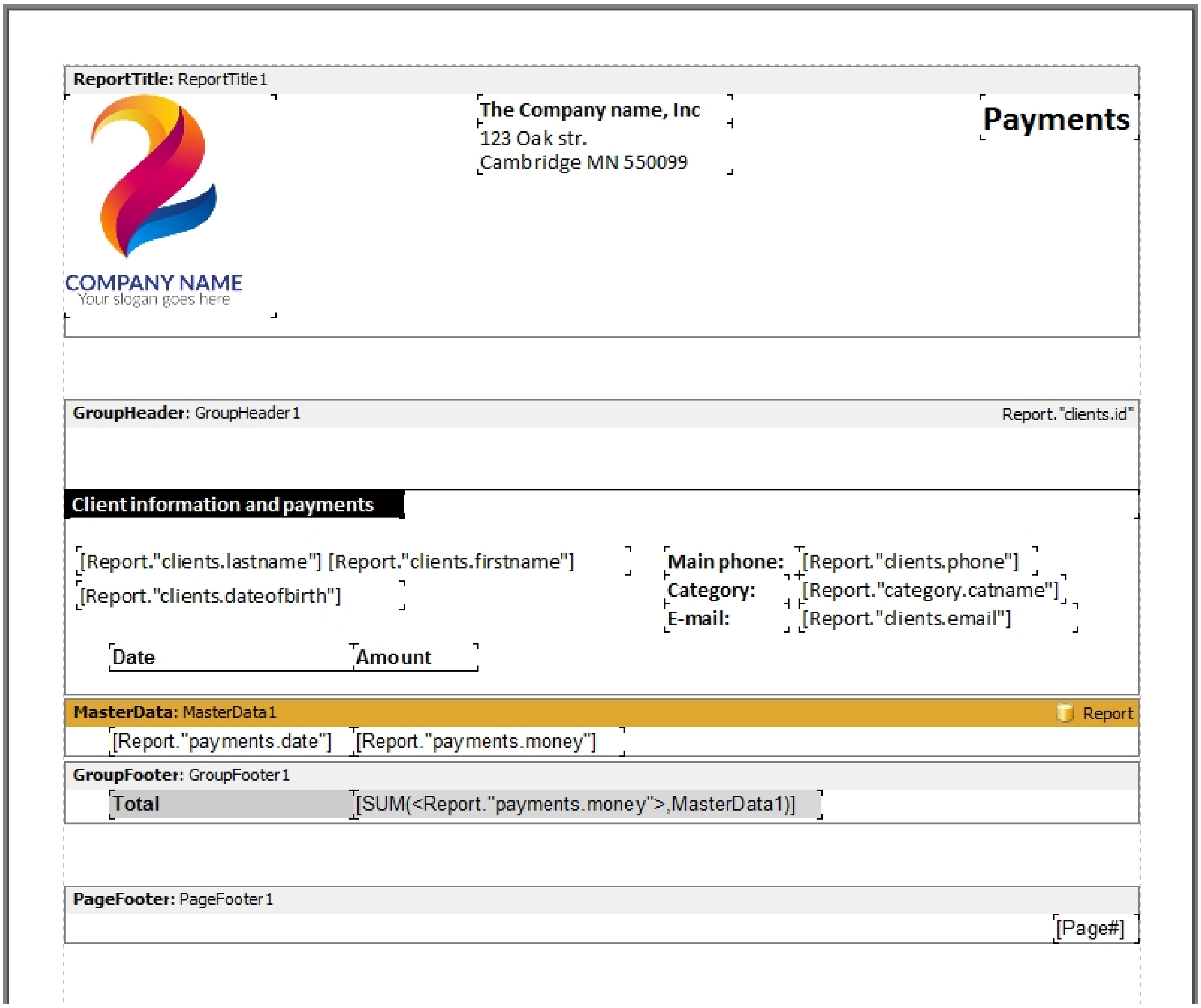
You can also apply monetary formatting to this field, as described in the previous step.

And the last block that we will place in the report is **Page Footer**, this block will be printed on each page of the report, we use it to numerate the pages.

Place the **Page#** system variable in this block from the right side of the report designer (Variables tab), just drag and drop:



Now your report template is fully ready



ReportTitle: ReportTitle1

COMPANY NAME
Your slogan goes here

The Company name, Inc
123 Oak str.
Cambridge MN 550099

Payments

GroupHeader: GroupHeader1 Report."clients.id"

Client information and payments

[Report."clients.lastname"] [Report."clients.firstname"]
[Report."clients.dateofbirth"]

Main phone: [Report."clients.phone"]
Category: [Report."category.catname"]
E-mail: [Report."clients.email"]

Date Amount

MasterData: MasterData1 Report

[Report."payments.date"] [Report."payments.money"]

GroupFooter: GroupFooter1

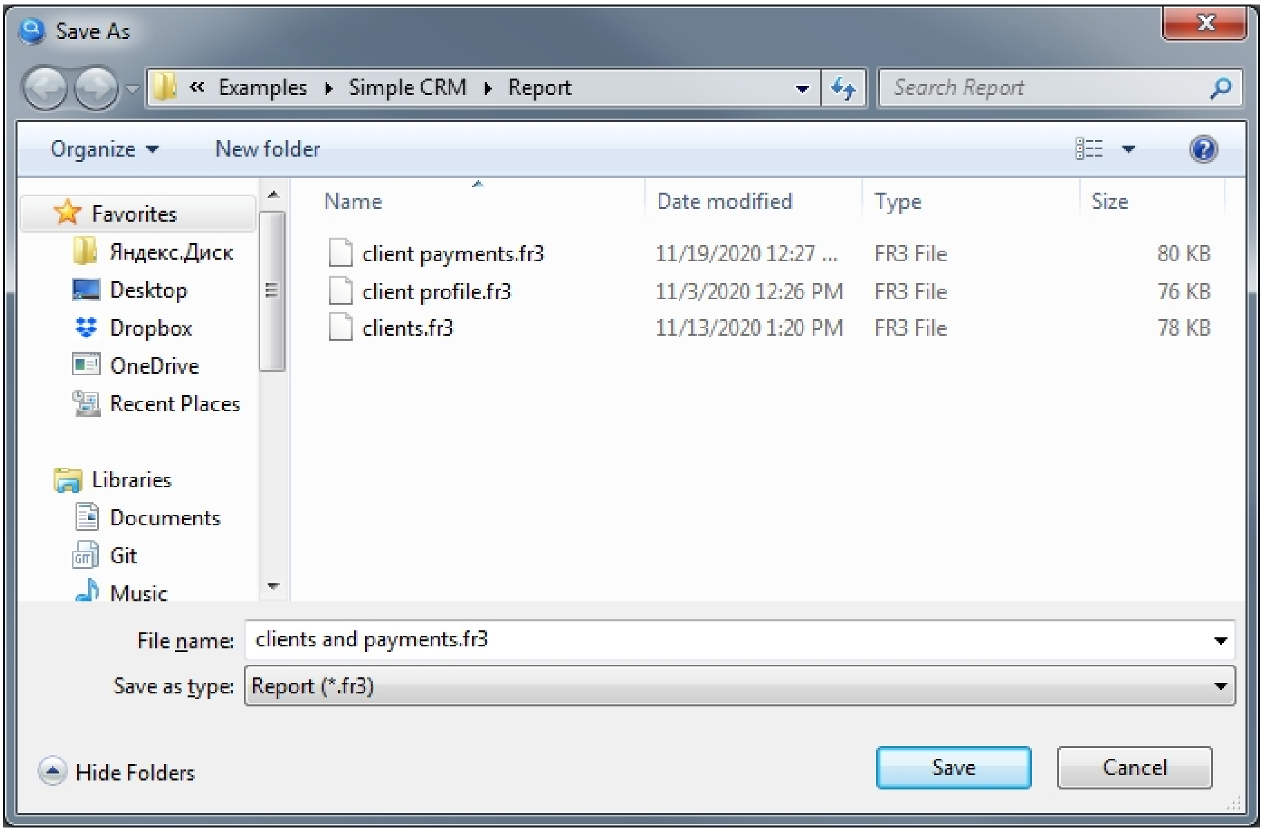
Total [SUM(<Report."payments.money">,MasterData1)]

PageFooter: PageFooter1

[Page#]

Go to the File > Preview menu or just press Ctrl+P to see what your report will look like.

Save the report template to the **Report** folder of your project, menu **File > Save As**... Give the file a name such as "**clients and payments**" as shown in the figure below:

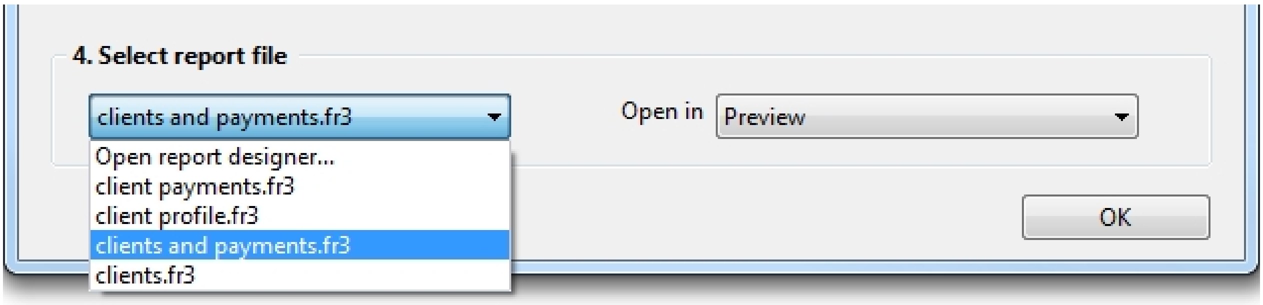


Name	Date modified	Type	Size
client payments.fr3	11/19/2020 12:27 ...	FR3 File	80 KB
client profile.fr3	11/3/2020 12:26 PM	FR3 File	76 KB
clients.fr3	11/13/2020 1:20 PM	FR3 File	78 KB

File name: clients and payments.fr3

Save as type: Report (*.fr3)

It remains to go back to the settings of this button and select this report template to use for printing:



clients and payments.fr3

Open in Preview

Here we go. Now you can start the project again. Click on this button and you will see the report ready to print.

Documentation on working with the report designer can be found here https://www.fast-report.com/public_download/UserManual-en.pdf

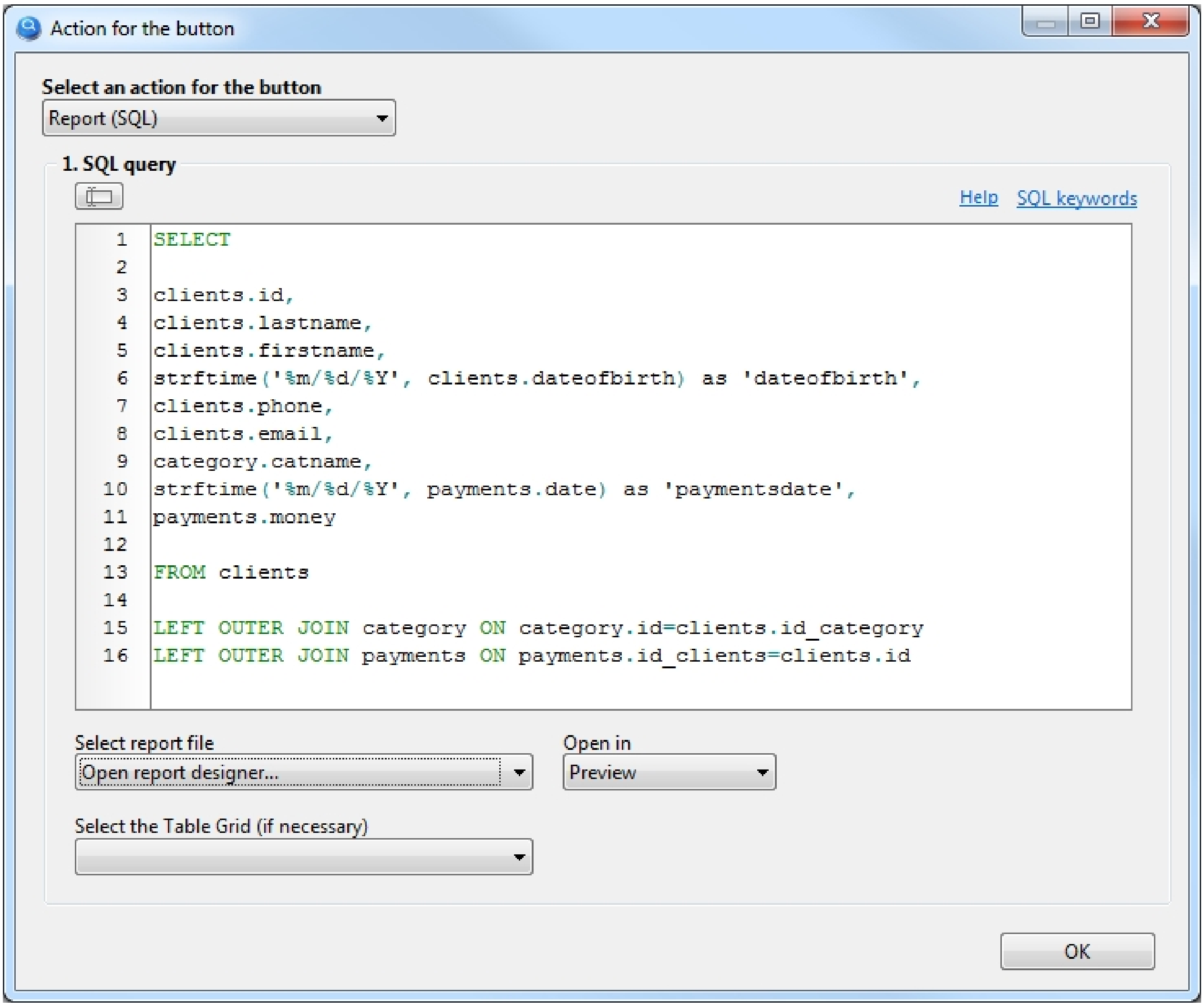
Report (SQL)

Description

Allows you to enter an SQL query for the report, and select the file defines the appearance and logic of the report.

**To work with reports, you must have knowledge of the query language SQL.*

Let us consider the principle of report setup. Let's say you need to print out all clients with their all payments.



Select an action for the button: Report (SQL)

1. SQL query

```
SELECT

clients.id,
clients.lastname,
clients.firstname,
strftime('%m/%d/%Y', clients.dateofbirth) as 'dateofbirth',
clients.phone,
clients.email,
category.catname,
strftime('%m/%d/%Y', payments.date) as 'paymentsdate',
payments.money

FROM clients

LEFT OUTER JOIN category ON category.id=clients.id_category
LEFT OUTER JOIN payments ON payments.id_clients=clients.id
```

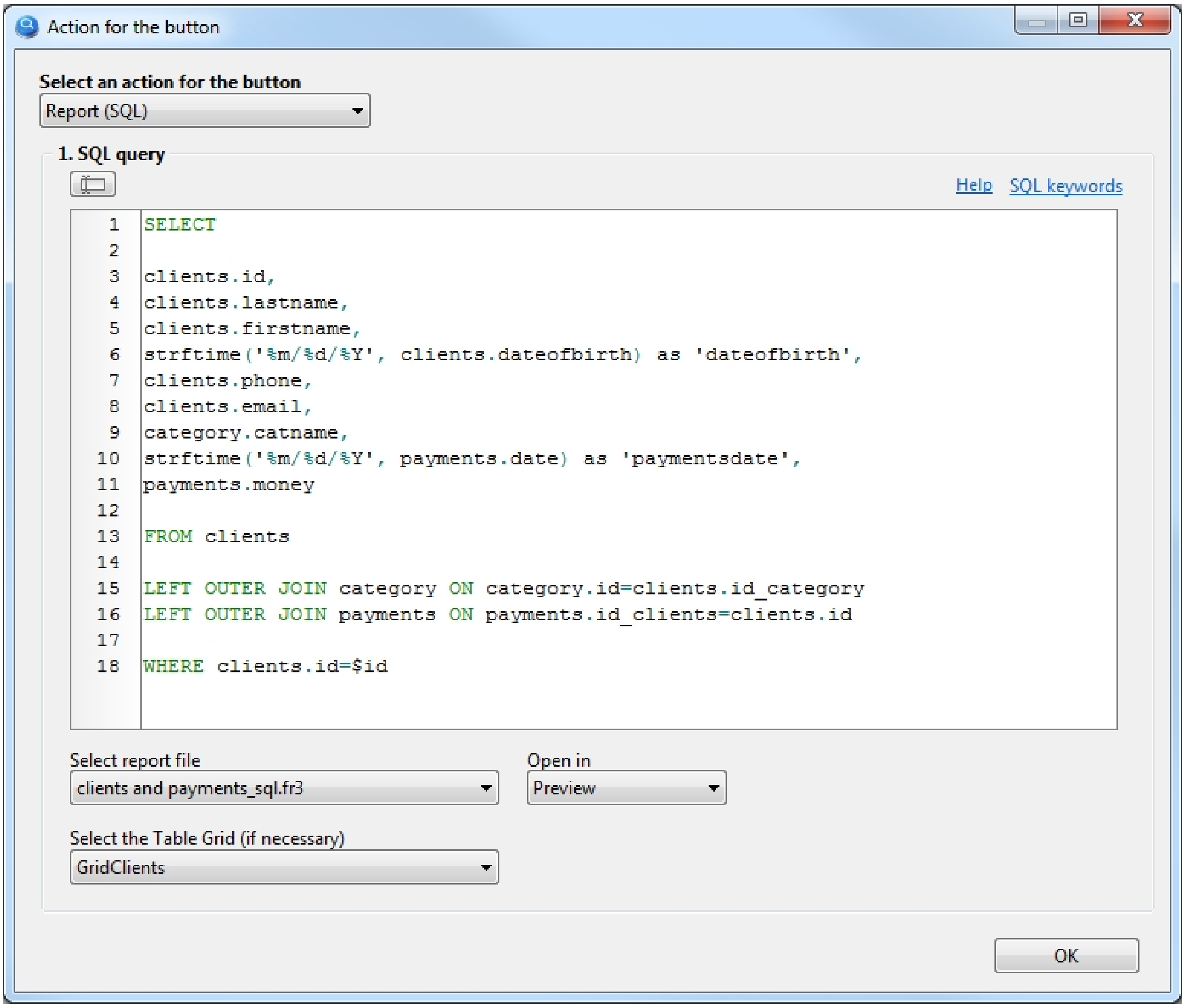
Select report file: Open report designer...

Open in: Preview

Select the Table Grid (if necessary):

1. Enter SQL query.
2. Since no template for this report has been created yet, let's leave the "*Select report file*" option as shown in the above figure.
3. Launch the project and click the button to which the "Report (SQL)" action has just been assigned.
4. You will see a report designer, where you need to create a report template, and save it to the "Report" folder, which is located in your project folder. You can read about how to work with the report designer in the Report section. The example "How to print a master-detail report with grouping" is most suitable for this SQL query. Also, detailed documentation on working with the report designer can be found here https://www.fast-report.com/public_download/UserManual-en.pdf
5. After you have created a report template, go back to the Action button property and select your template file from the "Select report file" list. Now when you click the button, you will see your finished report.

As you may have noticed, we have not yet used another option called "*Select the Table Grid*".



Select an action for the button: Report (SQL)

1. SQL query

```
SELECT

clients.id,
clients.lastname,
clients.firstname,
strftime('%m/%d/%Y', clients.dateofbirth) as 'dateofbirth',
clients.phone,
clients.email,
category.catname,
strftime('%m/%d/%Y', payments.date) as 'paymentsdate',
payments.money

FROM clients

LEFT OUTER JOIN category ON category.id=clients.id_category
LEFT OUTER JOIN payments ON payments.id_clients=clients.id

WHERE clients.id=$id
```

Select report file: clients and payments_sql.fr3

Open in: Preview

Select the Table Grid (if necessary): GridClients

This option allows you to obtain a record ID for your SQL query from the required table component. Let's say you need to print payments only from a client that you select from a TableGrid component. To do this, you need to add: *WHERE clients.id=$id* to your SQL query, where $id is the record identifier from the "clients" table, which will be substituted automatically.

The same result can be obtained if this button with the same settings is placed on the form for adding/editing the client, where the $id value is automatically assigned an identifier of the current record. In this case, leave the option "Select the Table Grid" empty.

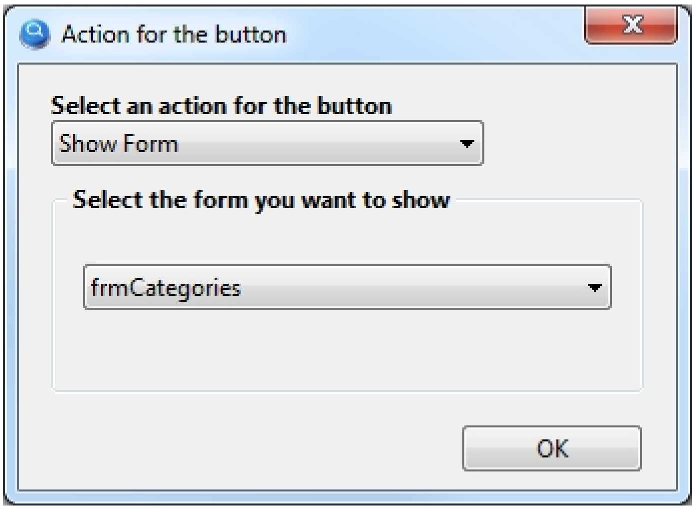
Additionally

If necessary, you can insert data from visual components into SQL query, read more about it in section SQL query.

Show form

Description

It is used to call the specified form on the screen.



Action for the button

Select an action for the button: Show Form

Select the form you want to show: frmCategories

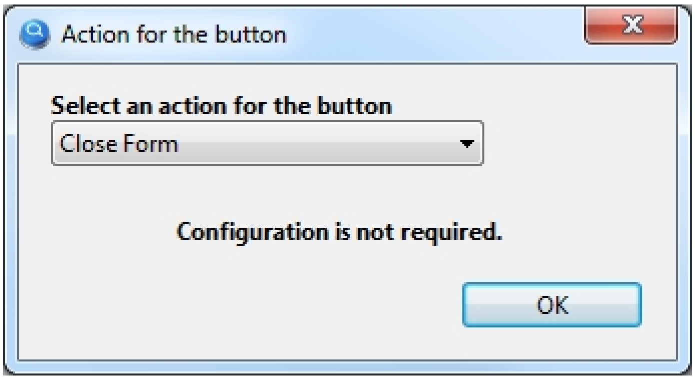
In this example, we call up a form that contains a list of categories to which the client can belong.

Do not use this action to create a new record.

Close form

Description

Closes the current form, i.e. the form on which this button is located.



Action for the button

Select an action for the button: Close Form

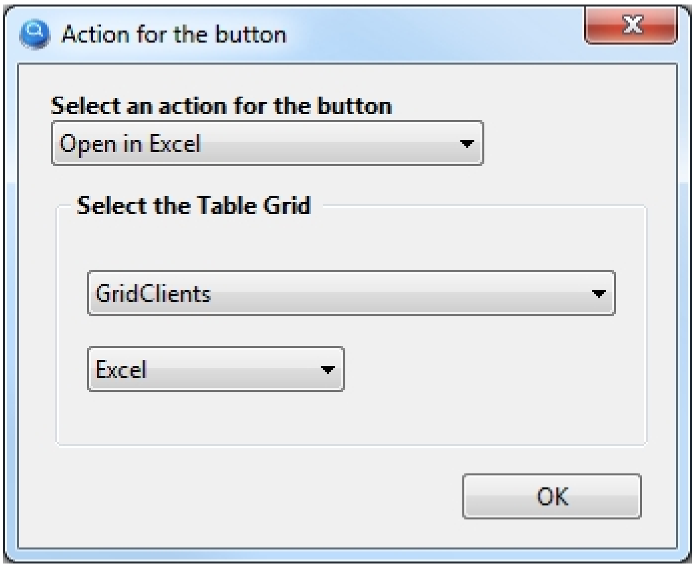
Configuration is not required.

This action does not need to be configured.
As a rule, this action is used for the Cancel button, not to save the record on the form, but simply to close it.

Open in Excel

Description

Opens the contents of a TableGrid component in Excel or Open Office.



Action for the button

Select an action for the button

Open in Excel

Select the Table Grid

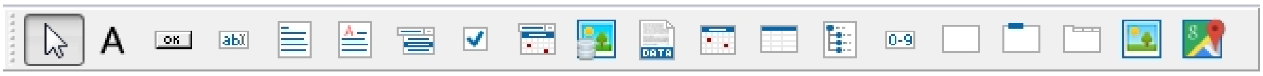
GridClients

Excel

It is necessary to select the TableGrid component, the content of which you want to open in Excel, when you click on the button.

User interface components

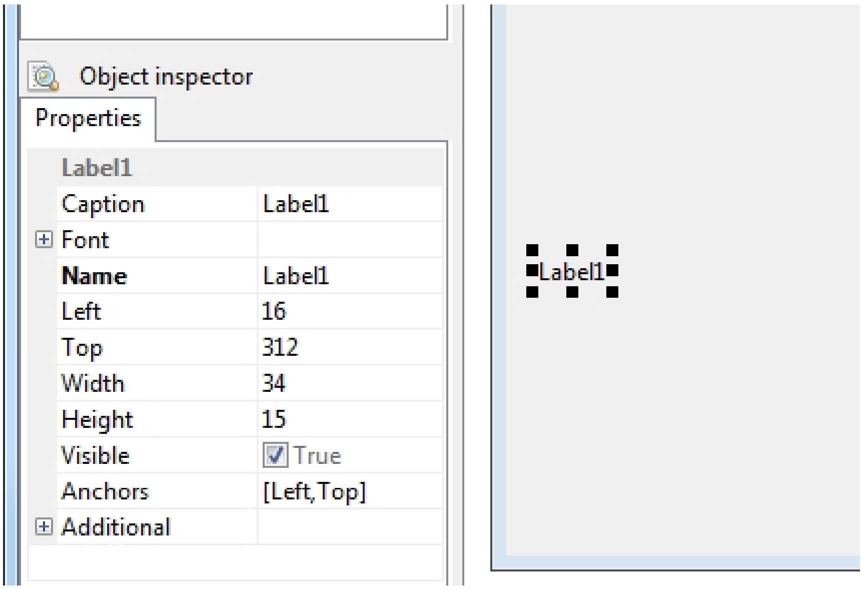
To create an interface, you have access to components that you can see on the toolbar.



	Название	Описание
A	**Label**	Label is a control that displays text on a form.
OK	**Button**	Button is a push button control. Use Button to put a standard push button on a form.
abI	**TextBox**	TextBox is a single-line edit control.
	Memo	Memo is a multiline edit control.
	RichEdit	Rich text edit controls let the user enter text that includes variation in font attributes, paragrap
	ComboBox	A ComboBox component is an edit box with a scrollable drop-down list attached to it.
✓	**CheckBox**	CheckBox represents a check box that can be on (checked) or off (unchecked).
	DateTimePicker	DateTimePicker is a component designed specifically for entering dates or/and times.

	DBImage	Allows saving the image to a database.
	DBFile	Allows saving a file to a database.
	Calendar	Calendar in which a user can select a date or range of dates.
	TableGrid	Allows outputting the records from the database in the table form.
	TreeView	TreeView displays a hierarchical list of item.
	Counter	Allows you to assign a unique number to records.
	Panel	Decorative interface element. It is a container for other components.
	GroupBox	Decorative interface element. It is a container for other components.
	PageControl	PageControl is a set of pages used to make a multiple page dialog box.
	Image	Allows you to place any picture on the form.
	Map	Allows you to place an interactive geographical map of Google Maps on the form, with the al

Each component has many properties that allow you to customize it to your needs. You can change the properties of a component in the "Object inspector" panel:



Label1

Caption	Label1
⊞ Font	
Name	Label1
Left	16
Top	312
Width	34
Height	15
Visible	☑ True
Anchors	[Left,Top]
⊞ Additional	

For convenience, the properties of all components are divided into two types: basic and additional. Basic properties are available immediately, when you select a component. To access additional properties, open the Additional section, which is located at the very end of the list of properties. As a rule, Additional properties are used less often than Basic properties.

Label

Description

Label is a control that displays text on a form. Use Label to add text that the user can't edit to a form.

Component properties

Property	Description
Caption	Specify the text string that labels the control.
Font → Color	Specifies the font color to use when displaying the text.
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
Alignment	Controls the horizontal placement of the text within the label.
AutoSize	Determines whether the size of the label automatically resizes to accommodate the text.
BiDiMode	Specifies the bi-directional mode for the component.
Color	Specifies the background color of the component. This property works if the Transparent property =
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by t
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The
Layout	Specifies the vertical placement of the text within the label.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on th
Transparent	Specifies whether controls that sit below the label on a form can be seen through the label.
WordWrap	Specifies whether the label text wraps when it is too long for the width of the label.

Button

Description

Button is a push button control. Use Button to put a standard push button on a form. An important and frequently used component. For example, it may serve to save a record to the database.

Performs a predefined action or a script when you click on it.

Component properties

Property	Description
Action	Allows you to select a button action. More info.
Caption	Specifies the text string that labels the control.
Font → Color	Specifies the font color to use when displaying the text. Starting with Windows Vista, you cannot char
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Icon	Allows you to select an icon for the button.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing cc
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
Style	Allows you to select the style of the button.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
BiDiMode	Specifies the bi-directional mode for the component.
Cancel	Determines whether the button will be automatically pressed when the Escape key is pressed.
ComandLinkHint	Text displayed as hint below button caption for Command Link. It makes sense if the Style button prc
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The
ImageAlignment	Alignment of image on button. It makes sense if the icon is selected in the Icon property.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on tl
WordWrap	Specifies whether the button text wraps to fit the width of the component.

Edit

Description

It is used to enter numerical and text information. TextBox controls can also display text to the user. As a rule, this component is assigned to a specific database field through the **TableName** and **FieldName** properties.

Component properties

Property	Description
TableName	Determines which database table a component belongs to.
FieldName	Determines which field of the database table this component belongs to.
BgColor	Specifies the background color of the component.
DefaultValue	Allows you to set the default value when creating a new record.
Filter	Allows you to select the data filtering condition when using this component together with the button
Font → Color	Specifies the font color to use when displaying the text.
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Increm. Search	Allows you to select the button with the action "Search" or "SQL query", which will be automatically cl
NumbersOnly	Allows only numbers to be typed into the text edit.
Currency	Enables formatting for monetary values.
Accuracy	Specifies the number of digits after the decimal point.
Prefix	The text to be inserted before the value, for example, it could be a dollar sign: $
Suffix	The text to be inserted after the value.
ThousandSep.	Specifies whether a thousand separator will be shown, for example: 10,000.00
ReadOnly	Determines whether the user can change the text of the edit control.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
Alignment	Determines how the text is aligned within the text edit control.
EditMask	Allows you to create an input mask.
AutoSelect	Determines whether all the text in the edit control is automatically selected when the control gets focu
AutoSize	Determines whether the height of the edit control automatically resizes to accommodate the text.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BorderStyle	Determines whether the edit component has a single line border around the client area.
BiDiMode	Specifies the bi-directional mode for the component.
CharCase	Determines the case of the text within the edit component.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The
MaxLength	Specifies the maximum number of characters the user can enter into the edit component.
PasswordChar	Indicates the character, if any, to display in place of the actual characters typed in the component. Ty
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on th

TextHint	A hint or message to be displayed when the Text property is empty.
Text	Contains a text string associated with the component.

Filter

Purpose

Allows you to select the data filtering condition when using this component together with the button with the "Search" action.

Description

The following filters are available

Value	Description
=	Exact match.
%s%	Search for substring in a string.
s%	Finds any values that starts with s.
>	Greater than. For numbers only.
>=	Greater than equal to. For numbers only.
<	Less than. For numbers only.
<=	Less than equal to. For numbers only.
<>	Not equal to. For numbers only.

Memo

Description

Multiline edit boxes allow the user to enter more than one line of text. As a rule, this component is assigned to a specific database field through the **TableName** and **FieldName** properties.

Component properties

Property	Description
TableName	Determines which database table a component belongs to.
FieldName	Determines which field of the database table this component belongs to.
BgColor	Specifies the background color of the component.
DefaultValue	Allows you to set the default value when creating a new record.
Font → Color	Specifies the font color to use when displaying the text.
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
ReadOnly	Determines whether the user can change the text of the edit control.

Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
Alignment	Determines how the text is aligned within the text edit control.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BorderStyle	Determines whether the edit component has a single line border around the client area.
BiDiMode	Specifies the bi-directional mode for the component.
CharCase	Determines the case of the text within the edit component.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by
Enabled	Controls whether the component responds to mouse and keyboard events.
HideSelection	Determines whether the visual indication of the selected text remains when focus shifts to another co
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The
MaxLength	Specifies the maximum number of characters the user can enter into the edit component.
ScrollBars	Determines whether the memo component has scroll bars.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on th
Text	Contains a text string associated with the component.
WantReturns	Determines whether the user can insert return characters into the text.
WantTabs	Determines whether the user can insert tab characters into the text.
WordWrap	Determines whether the edit control inserts soft carriage returns so text wraps at the right margin.

RichEdit

Description

Rich text edit controls let the user enter text that includes variation in font attributes, paragraph formatting information, images, tables and etc. RTF (Rich Text Format) storage format is used. As a rule, this component is assigned to a specific database field through the **TableName** and **FieldName** properties.

Component properties

Property	Description
TableName	Determines which database table a component belongs to.
FieldName	Determines which field of the database table this component belongs to.
DefaultValue	Allows you to set the default value when creating a new record.

ReadOnly	Determines whether the user can change the text of the edit control.
Ruler	Defines the presence of a horizontal ruler.
ToolBar1	Allows you to customize the visibility of buttons for the first row of the toolbar.
ToolBar2	Allows you to customize the visibility of buttons for the second row of the toolbar.
ToolBar3	Allows you to customize the visibility of buttons for the third row of the toolbar
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Enabled	Controls whether the component responds to mouse and keyboard events.

CheckBox

Description

CheckBox represents a check box that can be on (checked) or off (unchecked). The user can check the box to select the option, or uncheck it to deselect the option. If necessary, the component can have three states, such as On, Off and Grayed, to do this, set the AllowGrayed component property to True.

As a rule, this component is assigned to a specific database field through the **TableName** and **FieldName** properties.

Component properties

Property	Description
TableName	Determines which database table a component belongs to.
FieldName	Determines which field of the database table this component belongs to.
AllowGrayed	Determines whether check box can be in a "grayed" state.
Caption	Specifies a text string that identifies the component to the user.
Font → Color	Specifies the font color to use when displaying the text. Starting with Windows Vista, you cannot char
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Increm.Search	Allows you to select the button with the action "Search" or "SQL query", which will be automatically cl
DefaultState	Allows you to set the default value when creating a new record.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co

Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
Alignment	Determines whether the check box label aligns to the left or to the right of the tick box.
BiDiMode	Specifies the bi-directional mode for the component.
Checked	Defines the component value.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on t
State	Indicates whether the check box is selected, deselected, or grayed.
WordWrap	Specifies whether the text wraps to fit the width of the component.

DateTimePicker

Description

DateTimePicker is designed specifically for entering dates or/and times. As a rule, this component is assigned to a specific database field through the **TableName** and **FieldName** properties.

Component properties

Property	Description
TableName	Determines which database table a component belongs to.
FieldName	Determines which field of the database table this component belongs to.
Calendar	Allows you to select the DateTimePicker or Calendar component. It makes sense if the Kind=Time pro
DefaultChecked	Allows you to set Checked value when creating a new record. It makes sense if the ShowCheckbox pr
Font → Color	Specifies the font color to use when displaying the text. Starting with Windows Vista, you cannot cha
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Increm.Search	Allows you to select the button with the action "Search" or "SQL query", which will be automatically cl
DateFormat	Specifies format in which the date is presented. For example, "12/31/2020" or "31 December 2020". I
DateMode	Determines the method of date selection used by the component. It makes sense if the property Kind
Filter	Allows you to select the data filtering condition when using this component together with the button
Format	Specifies format in which the date or time is presented. More info.
Kind	Allows you to select a date mode. Date - date only. Time - time only. DateTime - date and time simul
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.

Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BiDiMode	Specifies the bi-directional mode for the component.
CalAlignment	Determines the alignment of the drop-down calendar.
Checked	Indicates whether the check box next to the date or time is selected.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by t
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on t
ShowCheckbox	Displays a check box next to the date or time.

property Calendar

Purpose

Allows you to bind to a component, the second component of the DateTimePicker or Calendar.

Description

For example, you need to save to the database at the same time the date and time, for this you need to link the two components, one of which will be the main and display time, and the second slave, his name must be entered in the Calendar property of the main component, the component can be subordinate **DateTimePicker** with property Kind = Date, or the component **Calendar**.

A subordinate component, there is no need to define properties **TableName** and **FieldName**, this must be done only by the main component.

property Filter

Purpose

Allows you to select the data filtering condition when using this component together with the button with the "Search" action.

Description

The following filters are available

Value	Description
=	Exact date match.
>	Greater than.
>=	Greater than equal to.
<	Less than.
<=	Less than equal to.

Example

Let's say you need to output records for a certain period. To do this, place two DateTimePicker components on the form, select ">=" for the Filter property in one of them and "<=" in the other.

Do not forget to select these components in the settings of the button with the action "Search".

property Format

Purpose

Allows you to set your own date or time format.

Description

For example, to make the date look like: Monday, 7 December 20, use the following format: dddd, d MMMM yy

The following format characters are understood:
yy = The last two digits of the year (that is, 2020 would be displayed as "20").
yyyy = The full year (that is, 2020 would be displayed as "2020").
M = The one- or two-digit month number.
MM = The two-digit month number. Single-digit values are preceded by a zero.
MMM = The three-character month abbreviation (Jan).
MMMM = The full month name. (January)
d = The one- or two-digit day.
dd = The two-digit day. Single-digit day values are preceded by a zero.
ddd = The three-character weekday abbreviation.
dddd = The full weekday name.

h = The one- or two-digit hour in 12-hour format.

hh = The two-digit hour in 12-hour format. Single-digit values are preceded by a zero.
H = The one- or two-digit hour in 24-hour format.
HH = The two-digit hour in 24-hour format. Single-digit values are preceded by a zero.
m = The one- or two-digit minute.
mm = The two-digit minute. Single-digit values are preceded by a zero.
t = The one-letter AM/PM abbreviation (that is, AM is displayed as "A").
tt = The two-letter AM/PM abbreviation (that is, AM is displayed as "AM").

If the DateTimePicker component has the property Kind = DateTime, you can change the time format using the script:

```
procedure Form1_OnShow (Sender: TObject; Action: string);
begin
  Form1.DateTimePicker1.TimeFormat := 'HH:mm';
end;
```

Calendar

Description

Calendar is a component that displays the month calendar of the specified year. As a rule, this component is assigned to a specific database field through the **TableName** and **FieldName** properties.

Component properties

Property	Description
TableName	Determines which database table a component belongs to.
FieldName	Determines which field of the database table this component belongs to.
Filter	Allows you to select the data filtering condition when using this component together with the button
Increm.Search	Allows you to select the button with the action "Search" or "SQL query", which will be automatically cli
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
WeekNumbers	Specifies whether week numbers are shown to the left of the calendar.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
BiDiMode	Specifies the bi-directional mode for the component.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The

MaxDate	Indicates the maximum date to which users can scroll the calendar.
MaxSelectRange	Specifies the maximum number of days that can be selected. It makes sense if MultiSelect property =
MultiSelect	Specifies whether multiple dates can be selected on the calendar.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on th
ShowToday	Specifies whether today's date is shown below the calendar.
ShowTodayCircle	Specifies whether today's date is circled on the calendar.
WeekNumbers	Specifies whether week numbers are shown to the left of the calendar.

ComboBox

Description

The component is used to display/select the record. After you determine which column of the table refers this component through the Foreign Key properties and FieldName, the ComboBox will contain all the records from the database table of the selected field.

For example you have a database table containing a list of professions, after you define the properties of the component **Foreign Key** and **FieldName**, a ComboBox will contain all the professions from the database table.

Component properties

Property	Description
ForeignKey	Allows you to select a foreign key. If the foreign key is missing in the database structure, you can wri
FieldName	Specifies the field whose records will be displayed in the component. You can specify multiple fields:
BgColor	Specifies the background color of the component.
Filter	A filter where you can set the condition for filtering records. You can use calculated fields in curly bra
Font → Color	Specifies the font color to use when displaying the text.
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Increm. Search	Allows you to select the button with the action "Search" or "SQL query", which will be automatically cl
DropDownCount	Specifies the maximum number of items displayed in the drop-down list.
DefaultIndex	The sequence number of the record in the list, selected by default when a new record is created. If 0,
Searchable	Determines whether it will be possible to quickly search for a record directly in the component.
Sort	Determines whether the records in the component will be sorted alphabetically. To sort by another fi
Name	Specifies the name of the component.
ParentComboBox	Allows you to specify a parent ComboBox, needed to create linked lists. For example: Country > City.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
AutoComplete	Positions to matching list items as you type. Makes sense if the Style property = csDropDown
AutoWidth	Specifies whether the dropdown list automatically adjusts its width depending on its content.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BiDiMode	Specifies the bi-directional mode for the component.
CharCase	Determines the case of the text in the combo box.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered b
Enabled	Controls whether the component responds to mouse and keyboard events.
FirstEmptyItem	Specifies whether the first item in the list is an empty value. Used to select a NULL value.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. Th
MaxLength	Specifies the maximum number of characters the user can type into the edit portion of the combo
MultiSelect	Enables multiple selection of entries in the component. Applies when using the button with the "Sea
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily or
SortField	Allows you to select the field in the database, by which the records in the component will be sortec
SortOrder	Allows you to select the sort order, ascending or descending.
Style	Determines the display style of the combo box.
Text	Contains a text string associated with the control.
TextHint	Specifies the text that is displayed as a text watermark in the edit box of the combo box control.

property ParentComboBox

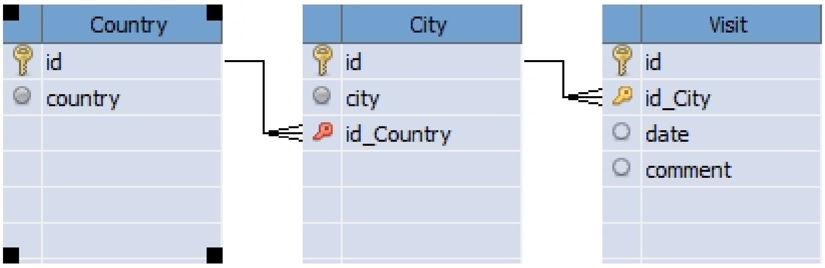
Purpose

It is necessary to create linked lists of two or more components.

Description

An example of a linked list is Countries and Cities. For example, when we select a country in the first ComboBox, and in the second linked ComboBox we can select only those cities that are present in the selected country.

To implement such a linked list, you must first create two tables in the database: **Country** and **City**.
The third table **Visit** is needed to record the visits of cities. Thus, the database structure will be as shown in the figure below:



Pay attention to the external key **id_Country** in the table **City**, as a rule, it is created with the option "Cascade delete", so when you delete any country from the table **Country**, cities belonging to the country will also be automatically removed.

Once the necessary tables in the database have been created, you can proceed to setting up components. In the **ParentComboBox** property of the ComboBox that shows cities, you must select the name of the ComboBox that shows countries. Thus, the ComboBox that shows cities becomes linked.

Here you can download an example project of this linked list.

TableGrid

Description

The component is used to output search results. Through the **Settings** property, the component can be configured to automatically output all records of the database table, usually used for dictionary tables and for the output of child records from the database table.

Additional features

While your project is running, this component has the following features.

1. You can copy the contents of a particular cell to the clipboard by holding down the Ctrl button and left-clicking on the cell.
2. You can copy the contents of the selected row to the clipboard by right-clicking and selecting "Copy row" from the menu that appears.
3. You can copy the entire contents of the table to the clipboard by right-clicking it and choosing "Copy all" or just Shift+Ctrl+C from the menu that appears. You can also paste the contents of the clipboard into Excel.

Component properties

Property	**Description**
Settings	Allows you to configure the component to show records from the database. More info.
Editable → AllowCreate	Responsible for the ability to create new records directly in the component.

Editable → AllowCreateEmpty	Allows the creation of new empty records.
Editable → AllowEdit	Enables editing records directly in the component.
Editable → AllowDelete	Enables the ability to delete entries directly in the component.
Editable → SecondClickEdit	Determines whether you need to double-click on a cell to edit a record.
Font → Color	Specifies the font color to use when displaying the text.
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
HeaderStyle	Allows you to select the header style for columns.
Increm. Search	Allows you to select the button with the action "Search" or "SQL query", which will be aut
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or c
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
AppearanceOptions	Configuring the appearance of the component. More info.
AutoScroll	Determines whether the scroll will be moved automatically to make the selected entry visible.
BiDiMode	Specifies the bi-directional mode for the component.
Caption	Allows you to write an caption on the component. The caption will disappear after the compon
Color	Specifies the background color of the component.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors prop
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covere
DefaultRowHeight	Defines the height of the rows.
Enabled	Controls whether the component responds to mouse and keyboard events.
EnableVisualStyles	Using visual styles for headings.
FixedCols	Specifies the number of fixed columns that will not be affected by horizontal scrolling.
FooterSize	Determines the height of the footer.
GridLinesColor	Defines the color of the lines that separate columns and rows.
GridLinesStyle	Defines the style of lines that separate columns and rows.
GridStyle	Allows you to select the style of the component. The gsSlides style requires the use of a script,
HeaderSize	Defines the height of the headers.
HideScrollBar	Determines whether the vertical scroll will be hidden when it is not needed.
HighlightedTextColor	Defines the color of the text in the selected row or cell.
Hint	Hint contains the text string that appears when the user moves the mouse over the component
HomeEndBehaviour	Defines the behavior of the Home and End buttons. hebTopBottom - move between the first ar
InactiveSelectionColor	Determines the background color of the selected row or cell when the component has no focu
InputSize	Defines the height of the zone to create a new record.
KeepUserSort	Determines whether the component will remember the column by which the user did the sortin
Limit	Allows you to limit the number of records that will be retrieved from the database.
MouseWheelEnabled	Determines whether the component will respond to mouse scrolling.
Options	Additional component settings. More info.
ReadOnly	Allows you to disable data editing in a component.

SelectionColor	Defines the color of the selected row or cell. Makes sense if the AppearanceOptions → aoAlph
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentaril
SortedStyle	Allows you to set the sorted column selection style.
WantTabs	Determines whether the Tab key will move the focus between cells within a component or whet

property AppearanceOptions

Purpose

Configuring the appearance of the component

Description

The following settings are available

Value	Description
aoAlphaBlendedSelection	Paint selection semi-transparent (alpha blended). This option mimic Windows Explorer SelectionColor property
aoBoldTextSelection	Bold text in selected cells.
aoHideFocus	Hide focus (dotted) rectangle.
aoHideSelection	Hide selection (painted in InactiveSelectionColor color) when grid is not focused.
aoHighlightSlideCells	Highlights the cell dimensions visually when the GridStyle component property = gsSlid
aoHintMarks	Control showing red triangles in top-left corner of cells with hint set. Form1.TableGric
aoIndicateSelectedCell	Additionally highlight the selected cell in the selected row.
aoIndicateSortedColumn	Paint sorted column in slightly dimmed color.

property Options

Purpose

Additional component settings

Description

The following settings are available

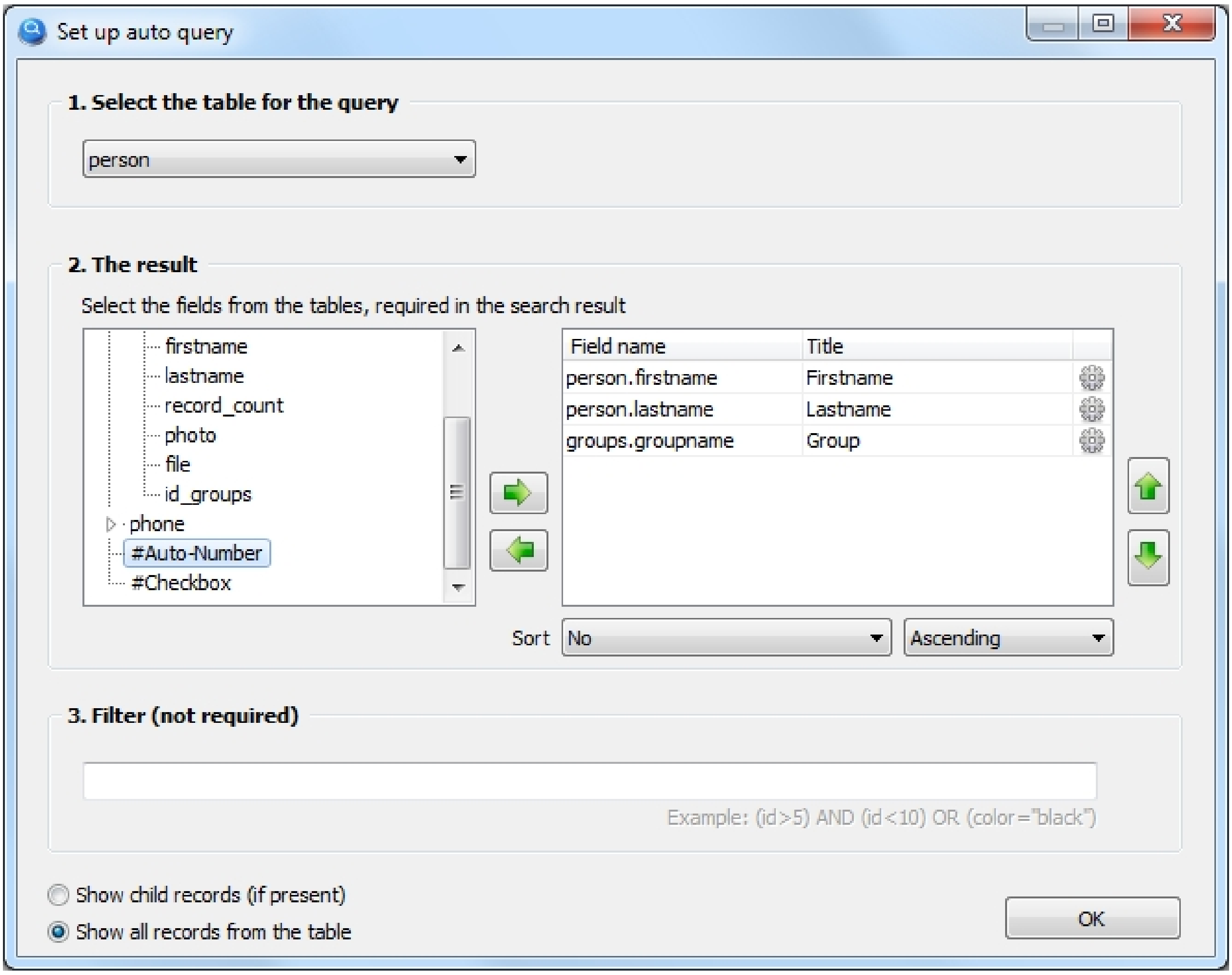
Value	Description
goArrowKeyExitEditing	When editing the contents of a cell, pressing the arrows on the keyboard exits the cell.
goCanHideColumn	Allows you to hide columns when dragging a column down. Makes sense if the goDisableC
goDisableColumnMoving	Prohibits dragging columns with the mouse.
goDisableKeys	Disables the ability to use the arrows on the keyboard.
goEscClearEdit	When editing the contents of a cell, pressing the Escape key on the keyboard will clear the
goFooter	Shows the footer.
goGrid	Shows the grid that separates columns and rows.
goHeader	Shows header.
goIndicator	Shows the indicator of the highlighted line.
goInput	Shows the input line below the column header.
goLockFixedCols	Prohibits drag and drop of fixed columns (see FixedCols).
goMultiSelect	Enables multiple row selection (using the Shift and Control buttons)
goRowResizing	Enables the ability to change the height of rows with the mouse. Makes sense if the goIndi
goRowMoving	Enables the ability to drag and drop rows with the mouse.
goSecondClickEdit	Editing a cell with a double click.
goSelectFullRow	Select the entire row.

property Settings

Purpose

Setting of the component is necessary when you want to display all records from the database table or child records on the record creation/editing form (for example, all phone numbers of the person).

Description



Set up auto query

1. Select the table for the query

person

2. The result

Select the fields from the tables, required in the search result

Field name	Title	
person.firstname	Firstname	⚙
person.lastname	Lastname	⚙
groups.groupname	Group	⚙

Sort: No, Ascending

3. Filter (not required)

Example: (id>5) AND (id<10) OR (color="black")

○ Show child records (if present)

● Show all records from the table

1. Select the database table from which we will take the information. In this example, we display a list of people, so choose a database table named **person**

2. Choose which fields of the database table you want to show. Note that we can add fields from other database tables, such as **groups.groupname**, because the table **person** has a foreign key to the table groups (in this example it is **person.ig_groups**).

We also assign names to the headers (First name, Last name, Group)

The third column with the icon ⚙ allows you to set the formula for calculating the total value in the footer and select the alignment of the text for that column. More info.

If necessary, you can choose which database field to sort by.

When selecting a database field to sort from a list, the final sorting is performed directly by the component itself, which increases the performance of retrieving data from the database. When specifying sorting manually (ORDER BY), sorting is performed on the DBMS side.

3. You can filter records by conditions. The syntax of the conditions is similar to the syntax of the SQL query language in the WHERE section. It is acceptable to use calculated fields in this field, which must be of the form: {tablename.calculated_field_name}

Note the line on the left side: **#Auto-Number**, you can use it to add a column with continuous numbering.

With the **#Checkbox** line, you can add a checkbox column, for example, to mark records you want. This column is usually used in conjunction with the script.

Note the options: "**Show child records**" and "**Show all records from table**".

If you need to display all records from the database table, select "**Show all records from table**".

If the component is on a form designed to create/edit a record and you need to show the child records, select "**Show child records**". An example of a child record can be all phone numbers that belong to the person.

Column setting

Purpose

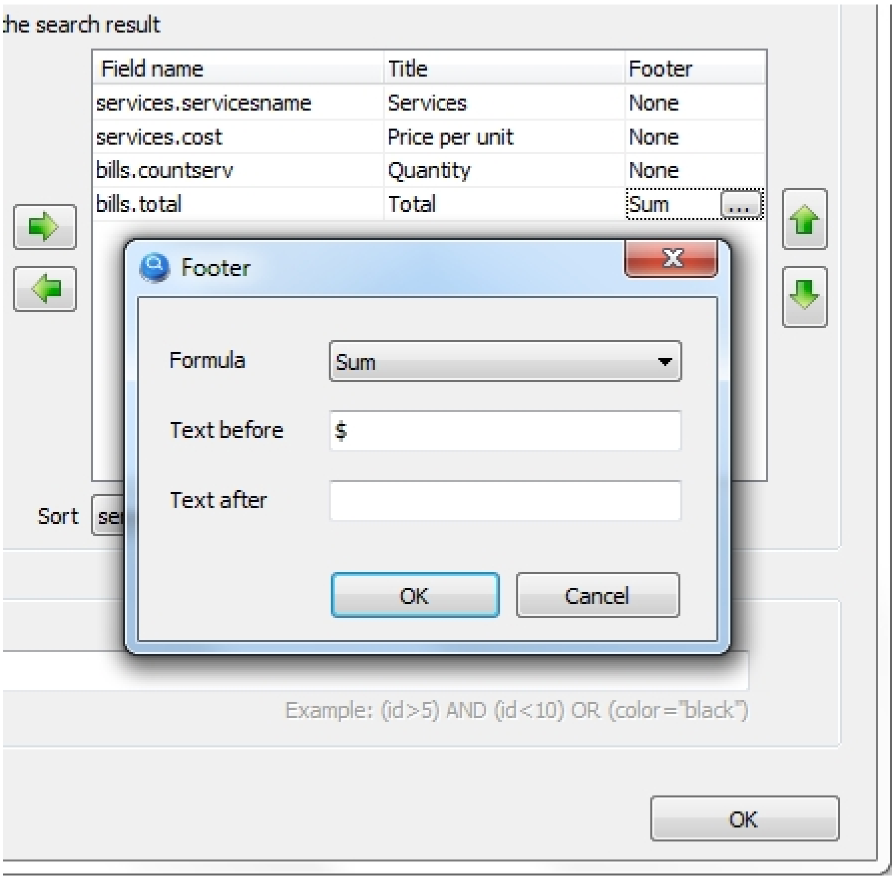
The column setting allows you to choose a formula to calculate the totals in the footer and set the text alignment.

Description

When configuring the button with the SEARCH action or configuring the TableGrid component, it is possible to select a formula to calculate the total in the desired column.

The following formulas are available to calculate the total:

- **none** - no formula
- **Sum** - sum calculation
- **Count** - row count
- **Average** - arithmetic mean
- **Maximum** - maximum value
- **Minimum** - minimum value
- **Distinct** - counting the number of unique values



he search result

Field name	Title	Footer
services.servicesname	Services	None
services.cost	Price per unit	None
bills.countserv	Quantity	None
bills.total	Total	Sum

Footer

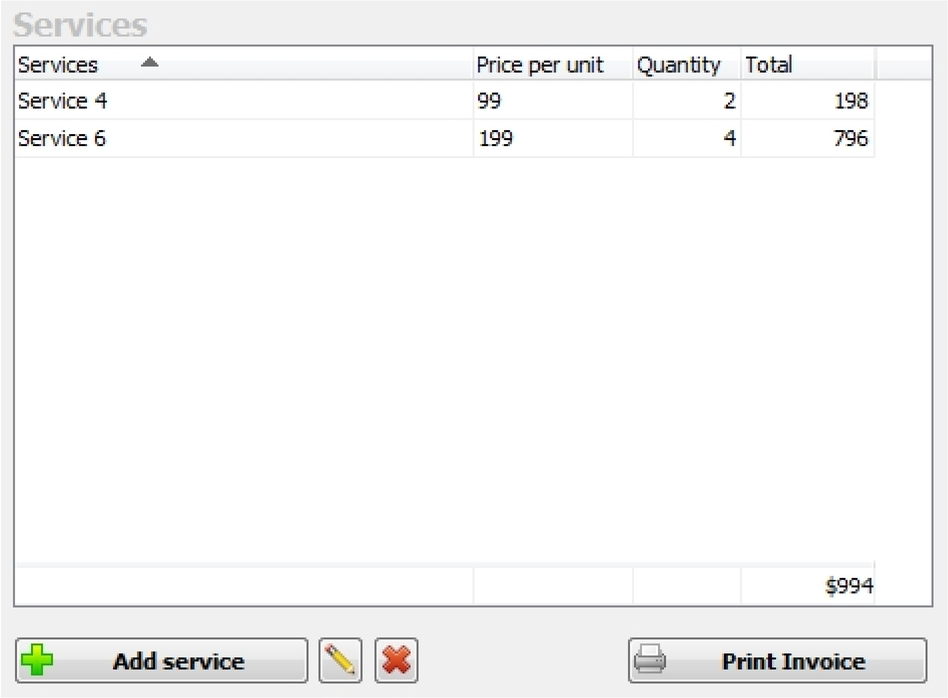
Formula: Sum

Text before: $

Text after:

Example: (id>5) AND (id<10) OR (color="black")

In this example, choose the formula Sum, which subtracts the sum of all values in the column. In addition, you can specify any text both before and after the total value.



Services

Services	Price per unit	Quantity	Total
Service 4	99	2	198
Service 6	199	4	796
			$994

The result of the setting, will be the total values in the footer of the table component.

Counter

Description

Allows you to assign a unique number to records. If you set the ReadOnly component property to False, you can edit the sequence number for the record if necessary, and this property is also used when this component is used for searching.

Unlike other components, there is no **FieldName** property, you only need to set **TableName**, which defines the name of the database table that contains a field with the type **COUNTER**.

Component properties

Property	Description
TableName	Determines which database table a component belongs to.
BgColor	Specifies the background color of the component.
Font → Color	Specifies the font color to use when displaying the text.
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
ReadOnly	Determines whether the user can change the text of the edit control.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
Alignment	Determines how the text is aligned within the text edit control.
AutoSelect	Determines whether all the text in the edit control is automatically selected when the control gets foc
AutoSize	Determines whether the height of the edit control automatically resizes to accommodate the text.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BorderStyle	Determines whether the edit component has a single line border around the client area.
BiDiMode	Specifies the bi-directional mode for the component.
CharCase	Determines the case of the text within the edit component.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by t
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on th

DBFile

Description

The component is used to save any file to the database*, with the ability to open or export it. The component can also link to a file or folder on your computer without directly saving it to the database.

This component is assigned to a specific database field via the **TableName** and **FieldName** properties.

** It is not recommended to save files directly to the database, as it causes it to slow down.*

Component properties

Property	**Description**
TableName	Determines which database table a component belongs to.
FieldName	Determines which field of the database table this component belongs to.
BgColor	Specifies the background color of the component.
CopyTo	Allows you to specify where you want the file to be copied automatically. More info.
Font → Color	Specifies the font color to use when displaying the text.
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Name	Specifies the name of the component.
Type	Sets the method of working with a file, save to database, link to file, link to folder.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	**Description**
Alignment	Determines how the text is aligned within the text edit control.
AutoSelect	Determines whether all the text in the edit control is automatically selected when the control gets foc
AutoSize	Determines whether the height of the edit control automatically resizes to accommodate the text.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BorderStyle	Determines whether the edit component has a single line border around the client area.
BiDiMode	Specifies the bi-directional mode for the component.
CharCase	Determines the case of the text within the edit component.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by t
Enabled	Controls whether the component responds to mouse and keyboard events.

HideSelection	Determines whether the visual indication of the selected text remains when focus shifts to another co
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The
MaxLength	Specifies the maximum number of characters the user can enter into the edit component.
ReadOnly	Determines whether the user can change the text of the edit control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on t
TextHint	A hint or message to be displayed when the Text property is empty.
Text	Contains a text string associated with the component.

CopyTo

Purpose

The **CopyTo** property allows you to specify where to automatically copy the file relative to the database file path. This property makes sense if the **Type** property has the value **LinkFile**.

Description

Possible values of the CopyTo property:

**** - file will be automatically copied to the folder with the database file

files - in the folder where the database file is located, files folder will be created, where the files will be automatically copied, it is allowed to specify a chain of folders, such as files\docs, these folders will be created automatically.

c:\files\ - the file will be automatically copied to the specified folder.

If the property is left empty, the file will not be copied automatically.

DBImage

Description

The component is used to save images to the database. Supported formats: jpg, bmp, gif, png8, png24. The component is assigned to a specific database field via the **TableName** and **FieldName** properties.

Component properties

Property	Description
TableName	Determines which database table a component belongs to.
FieldName	Determines which field of the database table this component belongs to.
CopyTo	Allows you to specify where you want the file to be copied automatically. More info.
Proportional	Determines whether the original aspect ratio of the image should be preserved if the **Stretch**=True p
Stretch	Determines whether the image will fit the dimensions of the component.

Transparent	Specifies whether the background of the image obscures objects below the image object.
Type	Sets the method of working with a file, save to database, link to file, link to folder.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
Autosize	Specifies whether the control sizes itself automatically to accommodate the dimensions of the imag
Center	Indicates whether the image is centered in the image control. Makes sense if the **Stretch** = False p
EnablePreview	Show the image in full size when you click on the component.
ShowButtons	Show buttons (Open, Save, Delete).
ShowButtonOpen	Show Open button.
ShowButtonSave	Show Save button.
ShowButtonDelete	Show Delete button.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered b
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. Th
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on

TreeView

Description

It serves for output and creation of data in a hierarchical form (tree structure). An example of hierarchical data is the structure of a company. The main configuration of the component is done through the **Settings** property.

Usage

Using the component, in many ways similar to using the ComboBox component. To use the TreeView component, you need to create a separate database table and a foreign key in another database table, which will store the user's selection. To use this component, you must also create an additional field in the database table (Parent ID), the field is required for the formation of records in a hierarchy, it automatically stores the identifier of the parent record.

Component allows you to create/edit/delete records, through context menu, without using additional forms and buttons (property **Editable**).

If necessary, you can create and edit records using a form, for this component has a **Form** property, in which you can select the form for these purposes.

Component properties

Property	Description
Settings	Allows you to configure the component to show records from the database. More info.
Editable → AllowCreate	Responsible for the ability to create new records directly in the component.
Editable → AllowEdit	Enables editing records directly in the component.
Editable → AllowDelete	Enables the ability to delete entries directly in the component.
Font → Color	Specifies the font color to use when displaying the text.
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Form	Allows you to select the form for creating/editing records.
HeaderStyle	Allows you to select the header style for columns.
Increm. Search	Allows you to select the button with the action "Search" or "SQL query", which will be auto
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or c
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
AppearanceOptions	Configuring the appearance of the component. Подробней.
AutoScroll	Determines whether the scroll will be moved automatically to make the selected entry visible.
BiDiMode	Specifies the bi-directional mode for the component.
Caption	Allows you to write an caption on the component. The caption will disappear after the compon
Color	Specifies the background color of the component.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors prop
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covere
DefaultRowHeight	Defines the height of the rows.
Enabled	Controls whether the component responds to mouse and keyboard events.
EnableVisualStyles	Using visual styles for headings.
ExpandLock	Allows you to prohibit the collapse of nodes.
FixedCols	Specifies the number of fixed columns that will not be affected by horizontal scrolling.
FooterSize	Determines the height of the footer.
GridLinesColor	Defines the color of the lines that separate columns and rows.
GridLinesStyle	Defines the style of lines that separate columns and rows.
GridStyle	Allows you to select the style of the component. The gsSlides style requires the use of a script.
HeaderSize	Defines the height of the headers.
HideScrollBar	Determines whether the vertical scroll will be hidden when it is not needed.
HighlightedTextColor	Defines the color of the text in the selected row or cell.
Hint	Hint contains the text string that appears when the user moves the mouse over the component

HomeEndBehaviour	Defines the behavior of the Home and End buttons. hebTopBottom - move between the first ar
InactiveSelectionColor	Determines the background color of the selected row or cell when the component has no focus
InputSize	Defines the height of the zone to create a new record.
KeepUserSort	Determines whether the component will remember the column by which the user did the sortin
MouseWheelEnabled	Determines whether the component will respond to mouse scrolling.
Options	Additional component settings. More info.
ReadOnly	Allows you to disable data editing in a component.
SelectionColor	Defines the color of the selected row or cell. Makes sense if the AppearanceOptions → aoAlph
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily
ShowLines	Show lines in the tree.
SortedStyle	Allows you to set the sorted column selection style.
WantTabs	Determines whether the Tab key will move the focus between cells within a component or whet

property Settings

Purpose

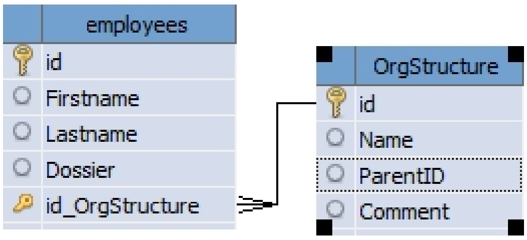
It serves for output and creation of data in a hierarchical form (tree structure). An example of hierarchical data is the structure of a company.

Description

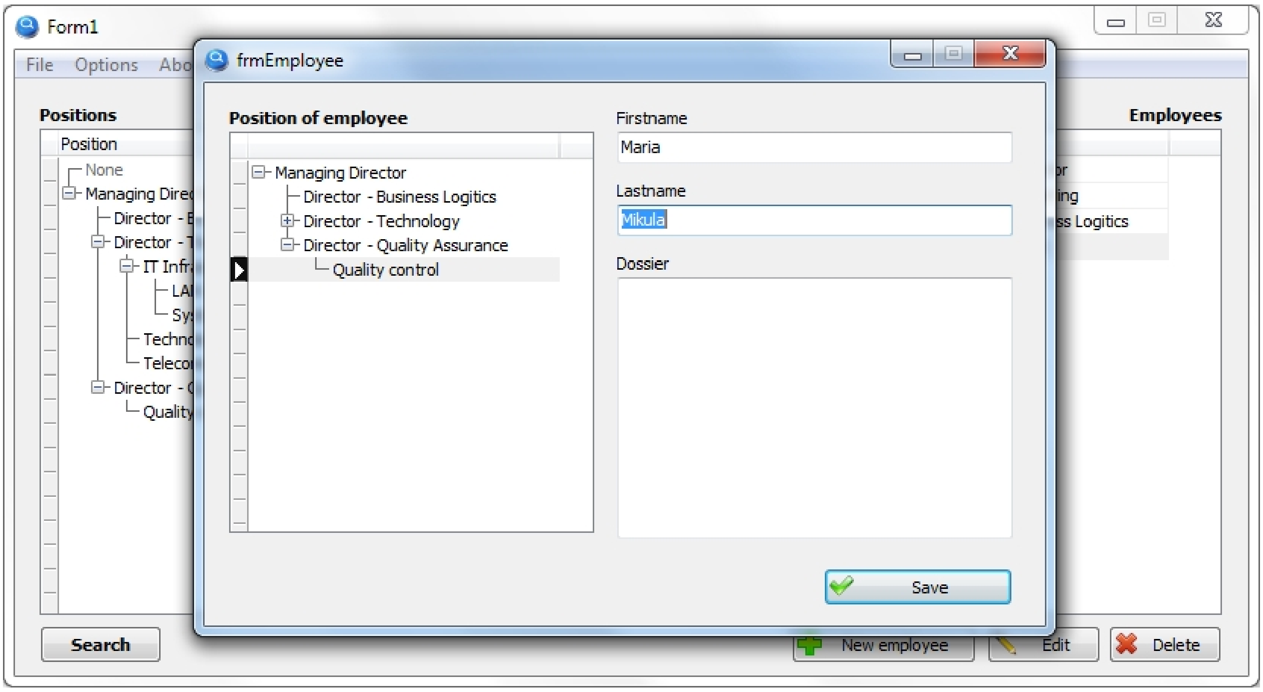
Using the component, in many ways similar to using the ComboBox component. To use the TreeView component, you need to create a separate database table and a foreign key in another database table, which will store the user's selection. To use this component, you must also create an additional field in the database table (Parent ID), the field is required for the formation of records in a hierarchy, it automatically stores the identifier of the parent record.

Let's look at an example where an employee can be assigned a position using the TreeView component.

The database structure of this example:



The form for creating/editing a record, where you can select a position for the employee will look like this:



Position of employee

- Managing Director
  - Director - Business Logitics
  - Director - Technology
  - Director - Quality Assurance
    - Quality control

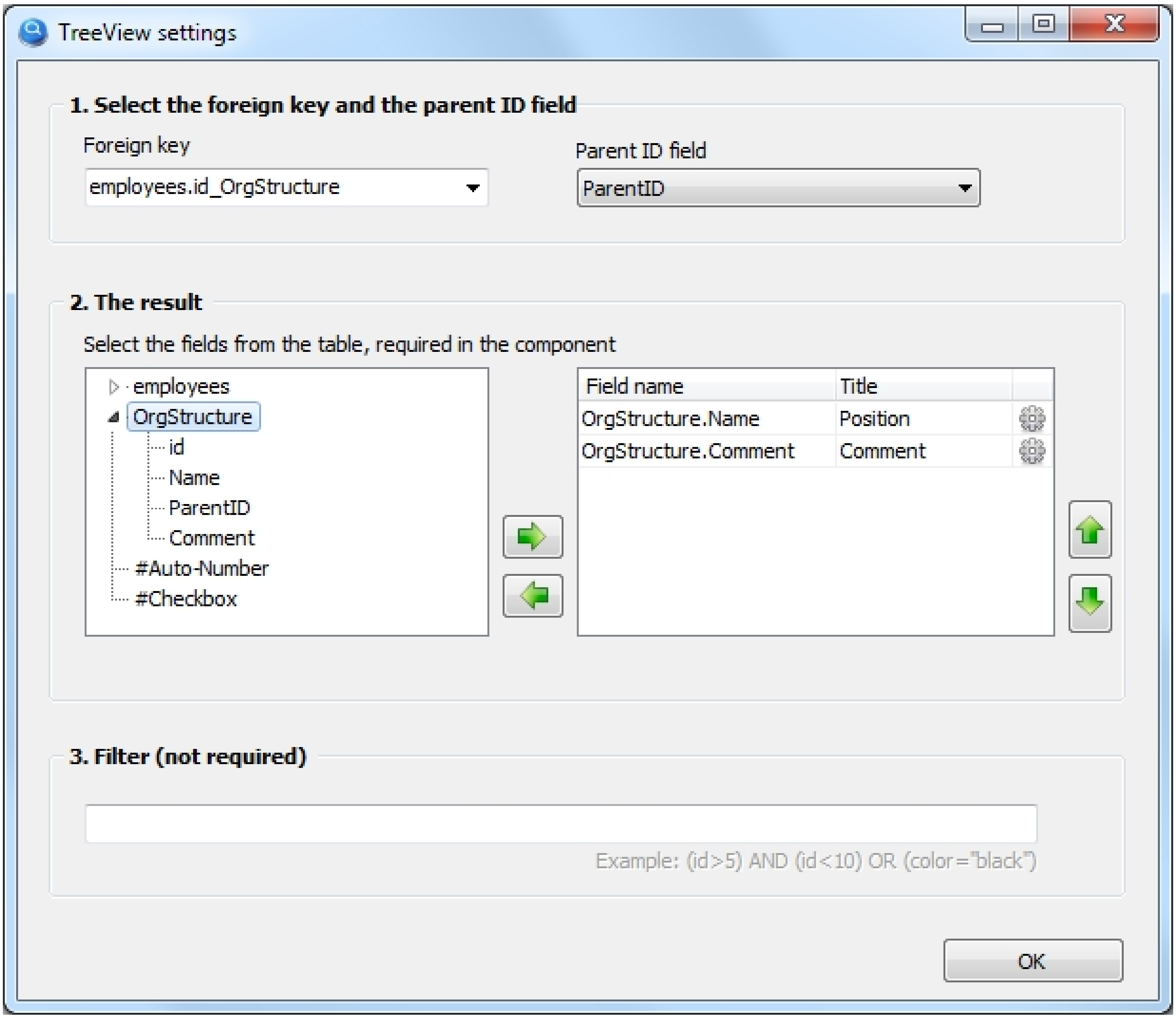
Firstname: Maria

Lastname: Mikula

Dossier

Save

The TreeView component is configured as follows:



1. Select the foreign key and the parent ID field

Foreign key: employees.id_OrgStructure

Parent ID field: ParentID

2. The result

Select the fields from the table, required in the component

- employees
- OrgStructure
  - id
  - Name
  - ParentID
  - Comment
- #Auto-Number
- #Checkbox

Field name	Title
OrgStructure.Name	Position
OrgStructure.Comment	Comment

3. Filter (not required)

Example: (id>5) AND (id<10) OR (color="black")

OK

1. Select an external key*, which will save the user's selection, similar to how it is saved in the ComboBox component. You should also select the field of "INTEGER" type, which will automatically save identifier, indicating the parent record. This is the field used by the system to form the hierarchy of records in the database table.

**Instead of a foreign key, you can write the table name, if the creation of a foreign key for this table is not planned in the database structure. I.e. instead of "employees.id_OrgStructure" you can write "OrgStructure"*

2. Choose which database table fields we need in the component.
In this example, we need a job position and a comment.

We also give names to the headers for the columns.

In the columns ⚙, you can set the formula, to calculate the totals and alignment. More info.

3. You can filter records by conditions. The syntax of the conditions is similar to the syntax of the SQL query language in the WHERE section. It is acceptable to use calculated fields in this field, which must be of the form: {tablename.calculated_field_name}

Here you can download the project with this example.

Map

Description

Allows you to place an interactive geographical map of Google Maps on the form, with the ability to put on the map markers, lines and polygons (placing lines and polygons is done using scripts.).

You can map one or more markers, then save their locations to the database. This component is assigned to two database fields via the **TableName** properties, **FieldLatitude** and **FieldLongitude**. The database fields to be used must be of the type "REAL".

Put a marker on the map through the context menu of a component (right mouse click) or by using the form on which you can place components to assign additional data to the marker (FormMarker property).

Component properties

Property	**Description**
TableName	Determines which database table a component belongs to.
FieldLatitude	Determines to which field of the database table belongs the latitude of the placed marker on the ma
FieldLongitude	Determines to which field of the database table belongs the longitude of the placed marker on the m
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.

Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
APIKey	Optionally specify an API Key to identify the application with the Google Maps API. Get an AP
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors pro
Enabled	Controls whether the component responds to mouse and keyboard events.
DefaultLatitude	Sets the latitude value for the default position.
DefaultLongitude	Sets the longitude value for the default position.
DefaultToCurrentLocation	Sets the current location as the default position. DefaultLatitude and DefaultLongitude are ig
DisableDoubleClickZoom	When set to true, disables zoom functions when double-clicking.
DisableMenu	Allows you to disable the context menu.
DisablePOI	When set to true, disable display of the points of interest on the map.
Draggable	When set to true, the entire map can be moved around in the control.
EnableKeyboard	When set to true, enables the use of the keyboard for controlling panning in the map (or in
FormMarker	Allows you to select the form to create/edit the marker, so you can specify additional inform
Language	Allows you to select the interface language on the map.
MapType	Sets the type of map (roadmap, hybrid, satellite, topographic)
MarkerIcon	Set the path to the image file to use as a marker icon. More info.
MarkersDraggable	Allows you to move markers on the map with the mouse.
MarkerInfoHTML	Allows you to set the text of the tooltip, which will be shown when you click on the marker. S
ScrollWheel	When set to true, enables the use of the scroll wheel. The scroll wheel can be used to zoom
ShowBicycling	When set to true, and if available in your country, bicycle trail information can be displayed
ShowTraffic	When set to true, and if available in your country, traffic information can be displayed.
ZoomMap	Is to be used to set the default zoom at startup. The zoom value is a value between 1 and 2

More information about putting markers on the map

Like other components, a map may contain information that can be stored in a database. Such information is the markers located on it. Thus, you can save to the database geographical coordinates of the object you need, which will be marked on the map as a marker.

To save the location of the marker on the map, you need to use two fields in the database simultaneously. Which is natural, because geographic coordinates consist of two parts, Latitude and Longitude.

Thus, to be able to put the marker on the map and save its location in the database, in the database table you need to create two fields with the type "REAL". Why is it a real number? Coordinates are represented as degrees, for example: 55.755831°, 37.617673°, which corresponds to this type of data.

You can map one or more markers. You can read more about this below.

Putting one marker on the map

In this case, using the component is no different from others. Just specify which database table and which database fields belong to this map using the **TableName**, **FieldLatitude** and **FieldLongitude** component properties. Then add this component to the list in the "**Save Record**" button settings.

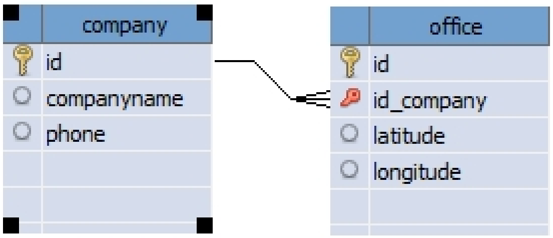
Putting more than one marker on the map

To be able to put several markers on the map and save their location, you need to create a separate database table, which will store records of these markers.

These markers will be child records and in this table you must have a foreign key to the parent table.

Putting several markers on the map can be compared to using the "**TableGrid**" component to work with child records, i.e. in our case markers will be child records.

An example of a database structure, when a company may have several offices and they need to be marked on the map.



MarkerIcon

Purpose

Set the path to the image file to use as a marker icon.

Description

As a graphic file, it is desirable to use an image in PNG format. You can specify the URL of the image file or a local file.

The marker URL must begin with http

When specifying a local file, you can specify either an absolute path (for example: c:\marker.png) or a relative path. For example marker.png, in this case this graphic file must be located in the folder of your project. Also, the file can be placed in a subfolder, for example: images\marker.png

MarkerInfoHTML

Purpose

Allows you to set the text of the tooltip, which will be shown when you click on the marker. Supports HTML and inserting data from the database.

Description

Consider an example of a hint:

```
<b>ID:</b> {id}<br>
<b>Office name:</b> {name}<br>
<b>Coordinates:</b> {latitude}; {longitude}<br>
<a href="http://google.com" target="_blank">Google.com</a><br>
<font color="red">This is some text!</font><br>
<img src="http://drive-software.com/images/logo.jpg" width="150">
```

Note the text surrounded by curly braces, e.g: {id}, {name}, etc.
Thus, the tooltip inserts the value of the fields from the database that belong to the marker.

FormMarker

Purpose

Allows you to select the form to create/edit the marker, so you can specify additional information for the marker.

Description

Creating a form that allows you to specify additional data for a marker is no different than creating a form for creating/editing a record.

When you put a marker on the map, the form specified in the FormMarker property will automatically be shown, which allows you to assign any necessary information to the marker. In addition to creating this form, you must create fields for this additional information in the same database table that stores the coordinates of the created marker.

Image

Description

Use Image to display a graphical image on a form. For example, you can place your company logo on the form. Supported formats are jpg, bmp, gif, png8, png24.

Component properties

Property	Description
Picture	Specifies the image that appears on the image control. The picture is automatically saved to the proj
Proportional	Indicates whether the image should be changed, without distortion, so that it fits the bounds of the in **Stretch=True** property
Stretch	Indicates whether the image should be changed so that it exactly fits the bounds of the image contro
Transparent	Specifies whether the background of the image obscures objects below the image object.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing co
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
AutoSize	Specifies whether the control sizes itself automatically to accommodate the dimensions of the image.
Center	Indicates whether the image is centered in the image control.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by t
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The h
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on th

PageControl

Description

PageControl is a set of pages used to make a multiple page dialog box.
Use PageControl to create a multiple page dialog or tabbed notebook. PageControl displays multiple overlapping pages that are TabSheet objects. The user selects a page by clicking the page's tab that appears at the top of the control.

Component properties

Property	Description
Style	Specifies the style of the tab control.
TabPosition	Determines whether tabs appear at the top or bottom.
Font → Color	Specifies the font color to use when displaying the text. Starting with Windows Vista, you cannot char
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing cc
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
BiDiMode	Specifies the bi-directional mode for the component.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The
HotTrack	Determines whether labels on the tab under the mouse are automatically highlighted.
Multiline	Determines whether the tabs can appear on more than one row.
RaggedRight	Specifies whether rows of tabs stretch to fill the width of the control.
ScrollOpposite	Determines how the rows of tabs are scrolled in a multi-line tab control. Makes sense if the Multiline
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on t
TabIndex	Identifies the selected tab on a tab control.

GroupBox

Description

The GroupBox component represents a standard Windows group box, used to group related controls on a form.

Component properties

Property	Description
Caption	Specifies a text string that identifies the control to the user.
Font → Color	Specifies the font color to use when displaying the text. Starting with Windows Vista, you cannot char
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing cc
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.
TabOrder	Indicates the position of the control in its parent's tab order.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
BiDiMode	Specifies the bi-directional mode for the component.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by t
Enabled	Controls whether the component responds to mouse and keyboard events.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The l
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on th

Panel

Description

Use Panel to put an empty panel on a form. Panels have properties for providing a beveled border around the control, as well as methods to help manage the placement of child controls embedded in the panel.

Component properties

Property	Description
BgColor	Specifies the background color of the control.
Name	Specifies the name of the component.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing cc
Width	Specifies the horizontal size of the control in pixels.
Height	Specifies the vertical size of the control in pixels.

TabOrder	Indicates the position of the control in its parent's tab order.
Visible	Specifies whether the component appears onscreen.
Anchors	Specifies how the control is anchored to its parent. More info.

Additional properties

Property	Description
AutoSize	Specifies whether the control sizes itself automatically to accommodate its contents.
BiDiMode	Specifies the bi-directional mode for the component.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BevelWidth	Determines the width, in pixels, of both the inner and outer bevels of a panel.
BorderStyle	Determines whether the edit component has a single line border around the client area.
BorderWidth	Specifies the distance, in pixels, between the outer and inner bevels.
Caption	Specifies a text string that identifies the control to the user.
Constraints	Specifies the size constraints for the component. It makes sense when using the Anchors property.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by t
Enabled	Controls whether the component responds to mouse and keyboard events.
Font → Color	Specifies the font color to use when displaying the text.
Font → Name	Identifies the typeface of the font.
Font → Size	Specifies the height of the font in points.
Font → Style	Determines whether the font is normal, italic, underlined, bold, and so on.
Hint	Hint contains the text string that appears when the user moves the mouse over the component. The I
ShowCaption	Specifies whether to display the caption of the panel control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on th
VerticalAlignment	Sets the vertical position of the caption.

property Anchors

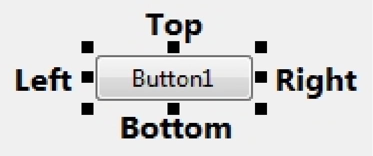
Purpose

Specifies how the control is anchored to its parent.

Description

Use Anchors to ensure that a control maintains its current position relative to an edge of its parent, even if the parent is resized. When its parent is resized, the control holds its position relative to the edges to which it is anchored.

The property is 4 switchable checkboxes: Left, Top, Right, Bottom.



By default, only **Left** and **Top** are set.

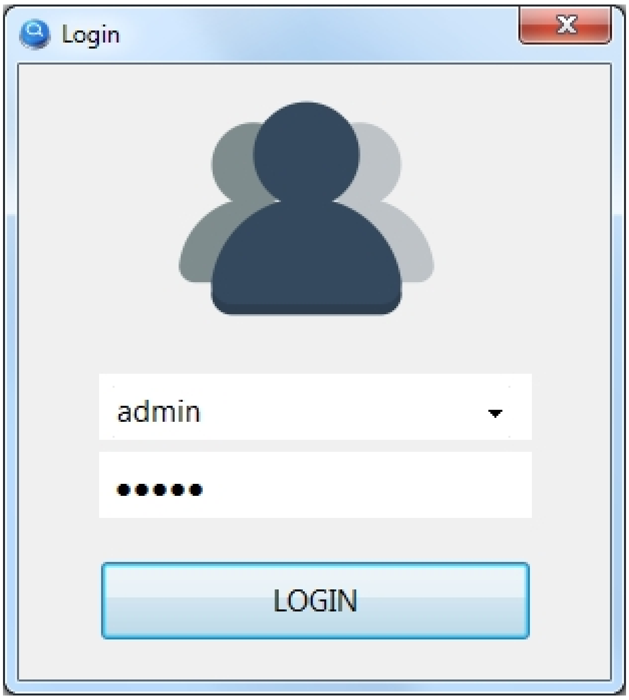
If a control is anchored to opposite edges of its parent, the control stretches when its parent is resized. For example, if a control has its Anchors property set to **Left**, **Right**, the control stretches when the width of its parent changes.

Access control

Introduction

Access control allows you to create a multi-user interface, where each group of users has access only to certain parts of the application or information.

- Setting up roles
- Setting up the user interface
- Setting up columns in the TableGrid component
- Access control to information
- Users creation

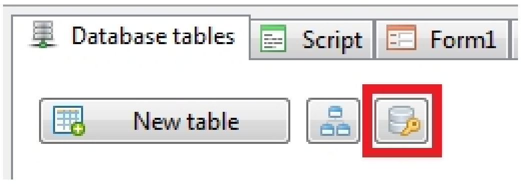


Setting up Roles

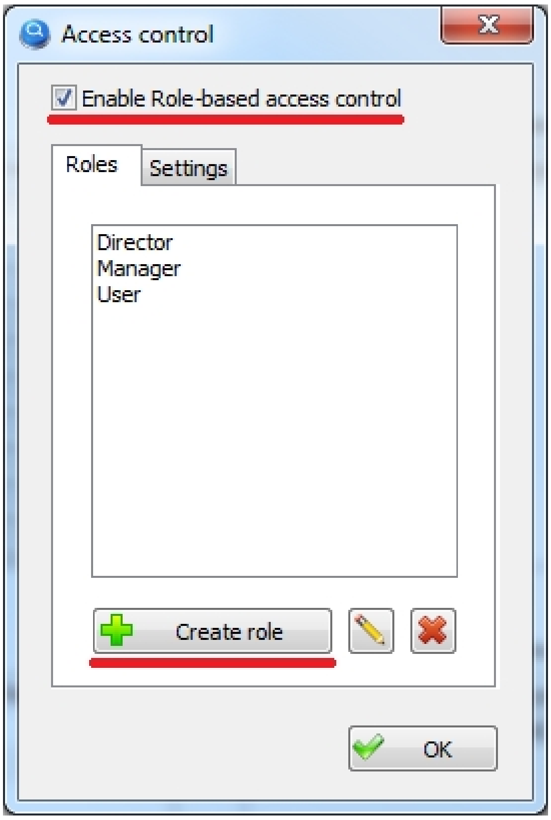
Setting up Roles

Access Control is used to create roles for users. You can therefore restrict access to certain buttons, input fields, table columns or hide certain information.

To enable access control in your project, click this button.



Check "Enable Role-based access control" and create the necessary roles.



Two tables in the database will be automatically created: "_user" and "_role". These tables are system tables and as a rule you will not have to interact with these database tables directly. You can only delete these tables after you have disabled Access Control for your project.

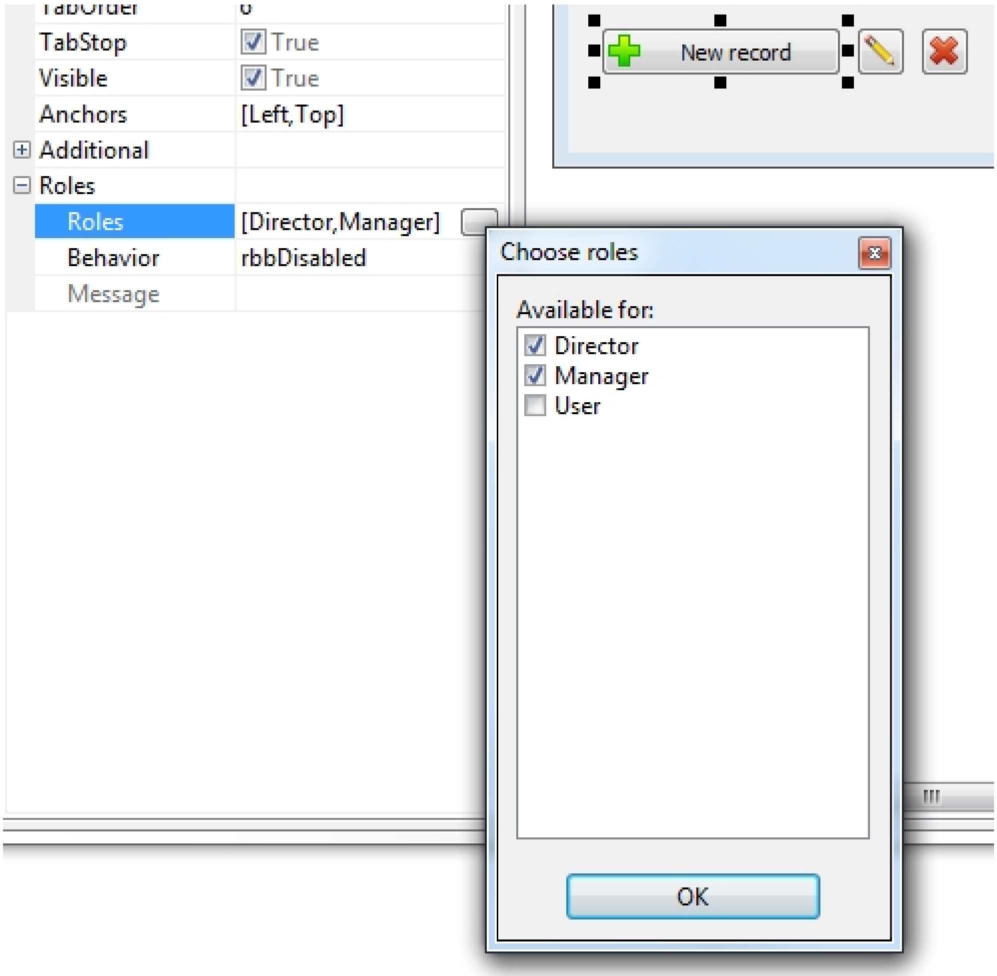
Next: Setting up the user interface

Setting up the user interface

Setting up the user interface

Once you have enabled and created roles, you can start configuring the interface for your project. For this, each component now has a "Roles" section.

Configure access for the "Button" component



TabOrder	0
TabStop	☑ True
Visible	☑ True
Anchors	[Left,Top]
⊞ Additional	
⊟ Roles	
Roles	[Director,Manager]
Behavior	rbbDisabled
Message	

Choose roles

Available for:

☑ Director
☑ Manager
□ User

In the "Roles" component property, select the roles. Thus, only users belonging to one of these roles will be able to access this component. If component roles are not selected, all users will have access to this component without any restrictions.

You can also choose how to restrict access to the component. The Behavior property is responsible for this, the following options are available:

- **rbbDisabled** - the component will be visible but inactive
- **rbbHide** - the component will be invisible
- **rbbShowMessage** - the component will be active. If the user does not have access to this button, instead of executing an action or script, he will receive a particular message in the "Message" property. If the message text is not specified in the "Message" property, the default message "Access denied" will be displayed.

The configuration of the remaining components is similar except for the "Behavior" property.

Properties "Behavior" of the components::

- **ribDisabled** - the component will be visible but inactive
- **ribHide** - the component will be invisible
- **ribReadOnly** - the component will be visible, but the option to edit the information it contains is disabled. You can copy text from the component to the clipboard.

Properties "Behavior" of the components:

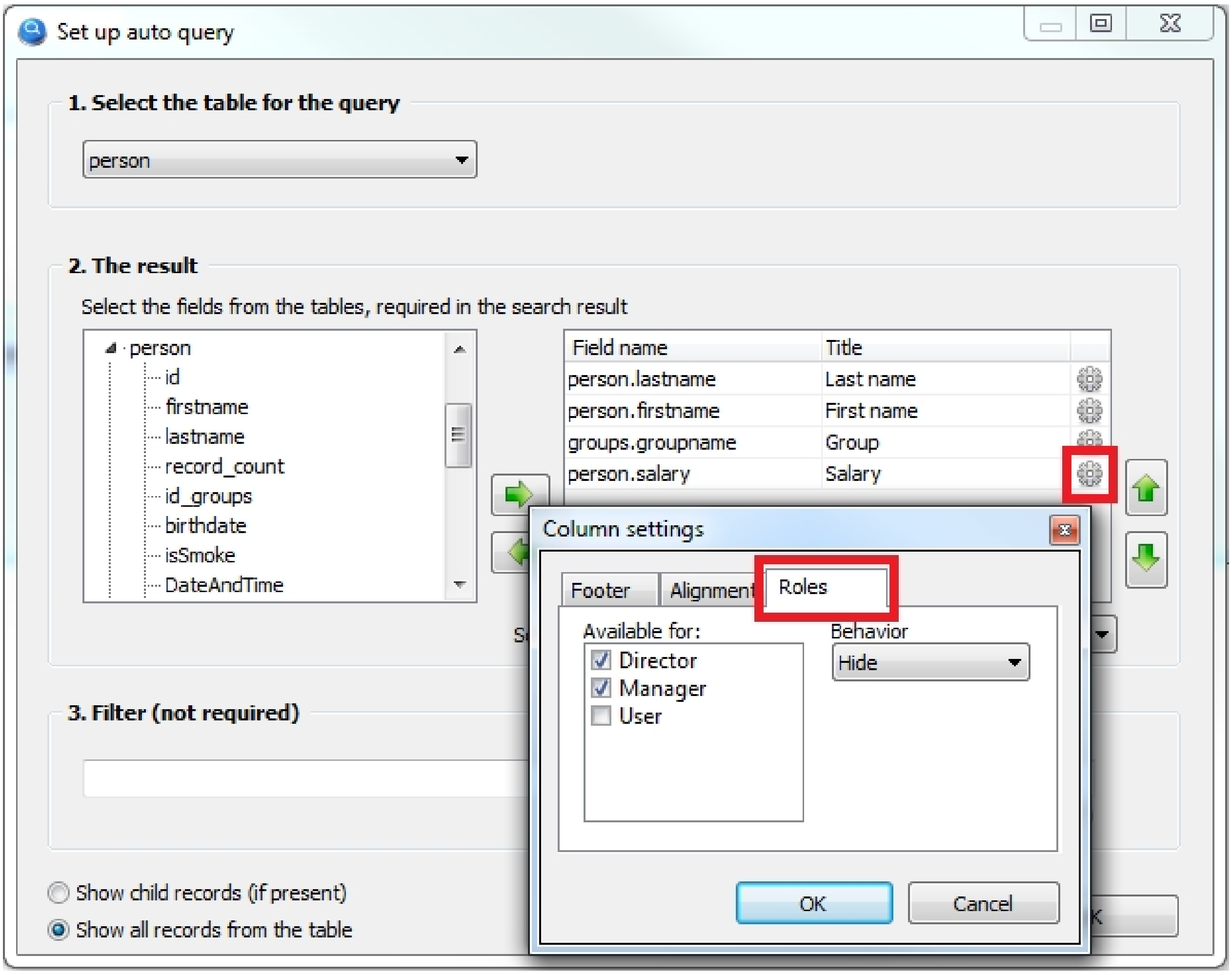
- **rsbDisabled** - the component will be visible but inactive
- **rsbHide** - the component will be invisible

Next: Setting up columns in the TableGrid component

Setting up columns in the TableGrid component

Setting up columns in the TableGrid component

Depending on the role of the user, you can hide the columns of this component.
Just select the roles for which this column will be available. If no role is selected, the column will be available to all users.



Field name	Title
person.lastname	Last name
person.firstname	First name
groups.groupname	Group
person.salary	Salary

Available for:
☑ Director
☑ Manager
☐ User

Behavior: Hide

From the "**Behavior**" drop-down list, you can choose "**Hide**" or "**Read Only**".

If you select Read Only, the column will not be hidden, but if the ability to edit data is enabled for this component (Editable property), then the ability to edit data for users who do not have the required role will be disabled for this column.

Next: Access control to information

Access Control to Information

Access Control to Information

In addition, you can hide entries in the components TableGrid and ComboBox, that the user should not see.

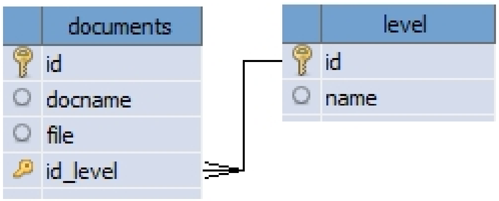
For this, the Roles component property has a **Data Filter** subkey in which you can write for each role a condition for filtering records. The syntax of the condition is similar to that of the SQL query language of the WHERE clause.

Example

Suppose you have a database of documents. Each document is given its importance: "High", "Medium", "Low". We make sure that:

- a user with the role "User" can only see documents with the importance of "Low".
- a user with the role "Manager" will see the documents with the importance of "Medium" and "Low".
- a user with the role "Director" will see all documents.

This is what the structure of the document database looks like and the data it contains.



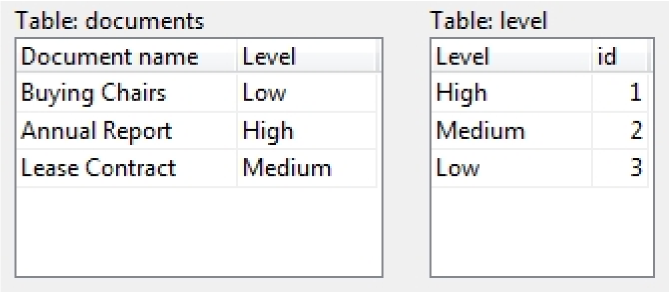


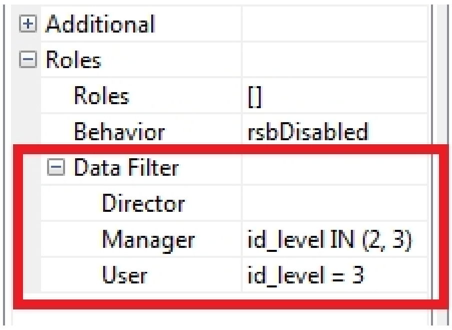
Table: documents

Document name	Level
Buying Chairs	Low
Annual Report	High
Lease Contract	Medium

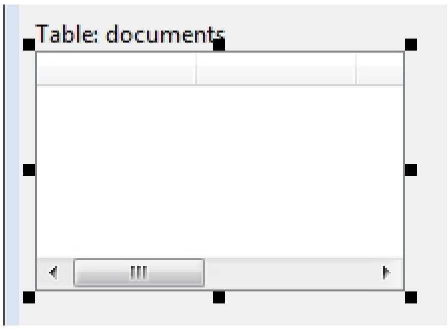
Table: level

Level	id
High	1
Medium	2
Low	3

Setting up the TableGrid component will look like this:



⊞ Additional	
⊟ Roles	
Roles	[]
Behavior	rsbDisabled
⊟ Data Filter	
Director	
Manager	id_level IN (2, 3)
User	id_level = 3

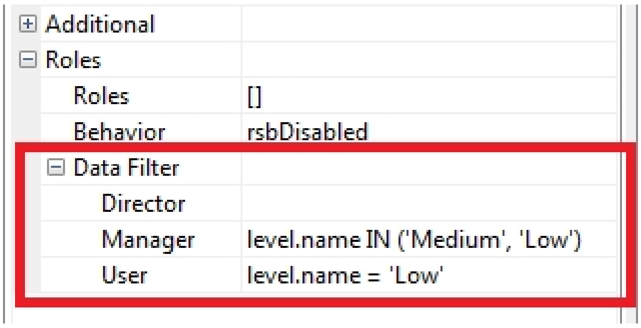


For the Director role we leave the field empty, so this role will have access to any documents.

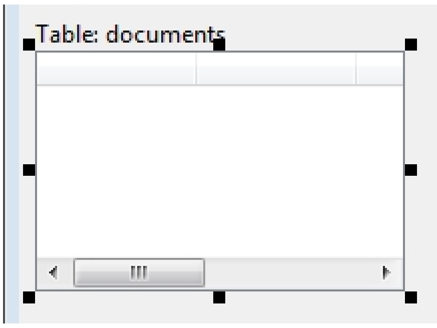
For the Manager role, list the document importance identifiers, so this role will have access to documents of Medium and Low importance.

For the User role, only documents with a Importance ID of 3 will be available, which corresponds to documents of Low importance.

This setting is also acceptable



⊞ Additional	
⊟ Roles	
Roles	[]
Behavior	rsbDisabled
⊟ Data Filter	
Director	
Manager	level.name IN ('Medium', 'Low')
User	level.name = 'Low'



But this has several disadvantages:

1. The field "level.name" must be present in the settings of the component "TableGrid"

2. If you change the type of importance, for example from "Low" to "Minor", you will need to correct the condition.

3. Work more slowly.

Users with the "Director" role can see all documents, so there is no requirement for data filtering.

The "ComboBox" component is configured in the same way.

Next: Users creation

Users creation

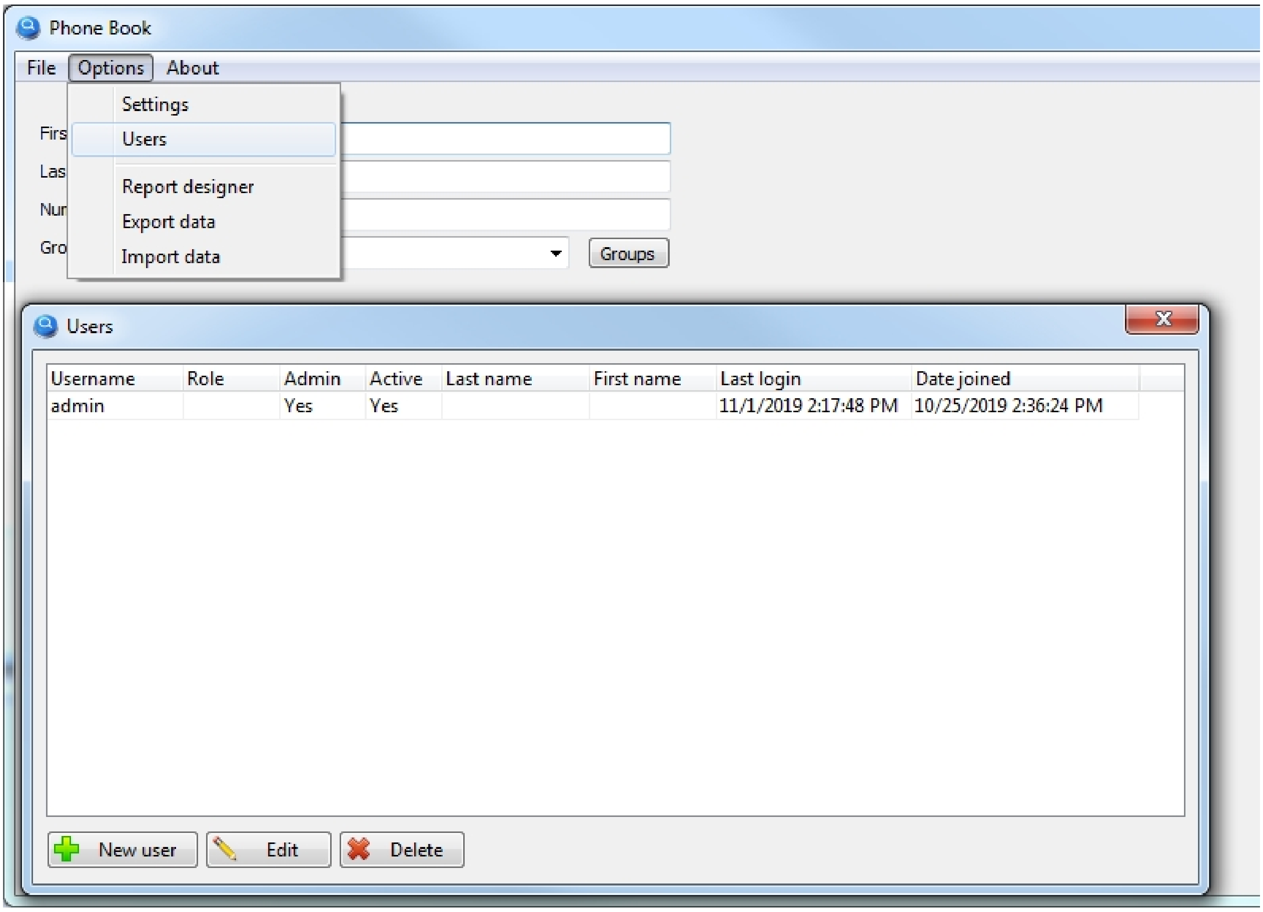
Users creation

After configuring the roles and user interface, you must create users. For each user, you must select his role, thus determining the actions he can perform in your program.

The first time you start your project, an administrator user is automatically created with username: admin, password: admin (don't forget to change the password).

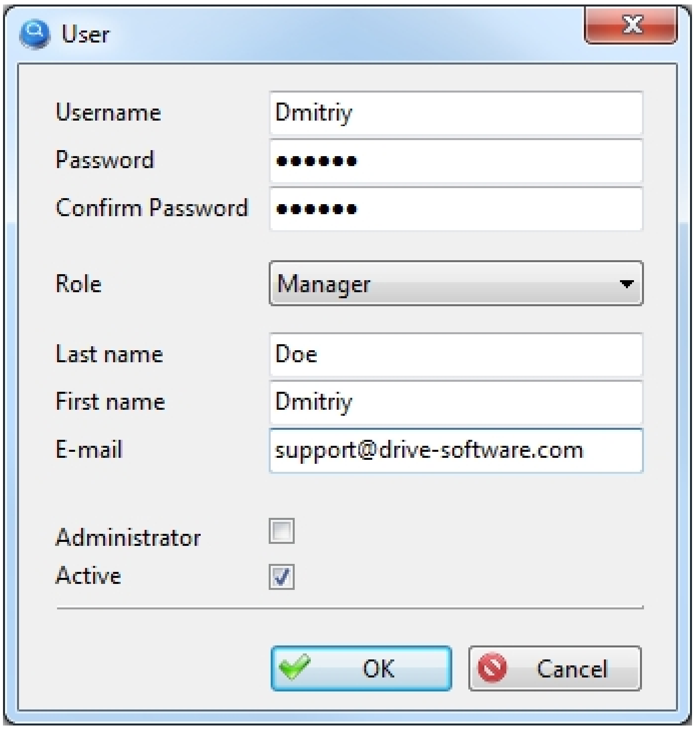
An administrator user is needed to create, modify or delete users. The administrator can give administrator rights to another user. Administrator has access to all functionality of your program without any limitations.

Creation of users is performed in the launched project through the menu "Options" > "Users", this menu is available only to users with administrator rights.



Username	Role	Admin	Active	Last name	First name	Last login	Date joined
admin		Yes	Yes			11/1/2019 2:17:48 PM	10/25/2019 2:36:24 PM

To create a new user, click on the "**New user**" button.



Username	Dmitriy
Password	••••••
Confirm Password	••••••
Role	Manager
Last name	Doe
First name	Dmitriy
E-mail	support@drive-software.com
Administrator	□
Active	☑

When creating a user, do not forget to choose their role, thus determining their ability to interact with your program.

Web access via browser

Webgrid

The application allows you to organize simple access to your database data via a browser. Only basic data operations, such as creating/editing/deleting records (CRUD) and search with output to a table will be available.

You can use this link to test this feature:
http://myvisualdatabase.com/webgrid/

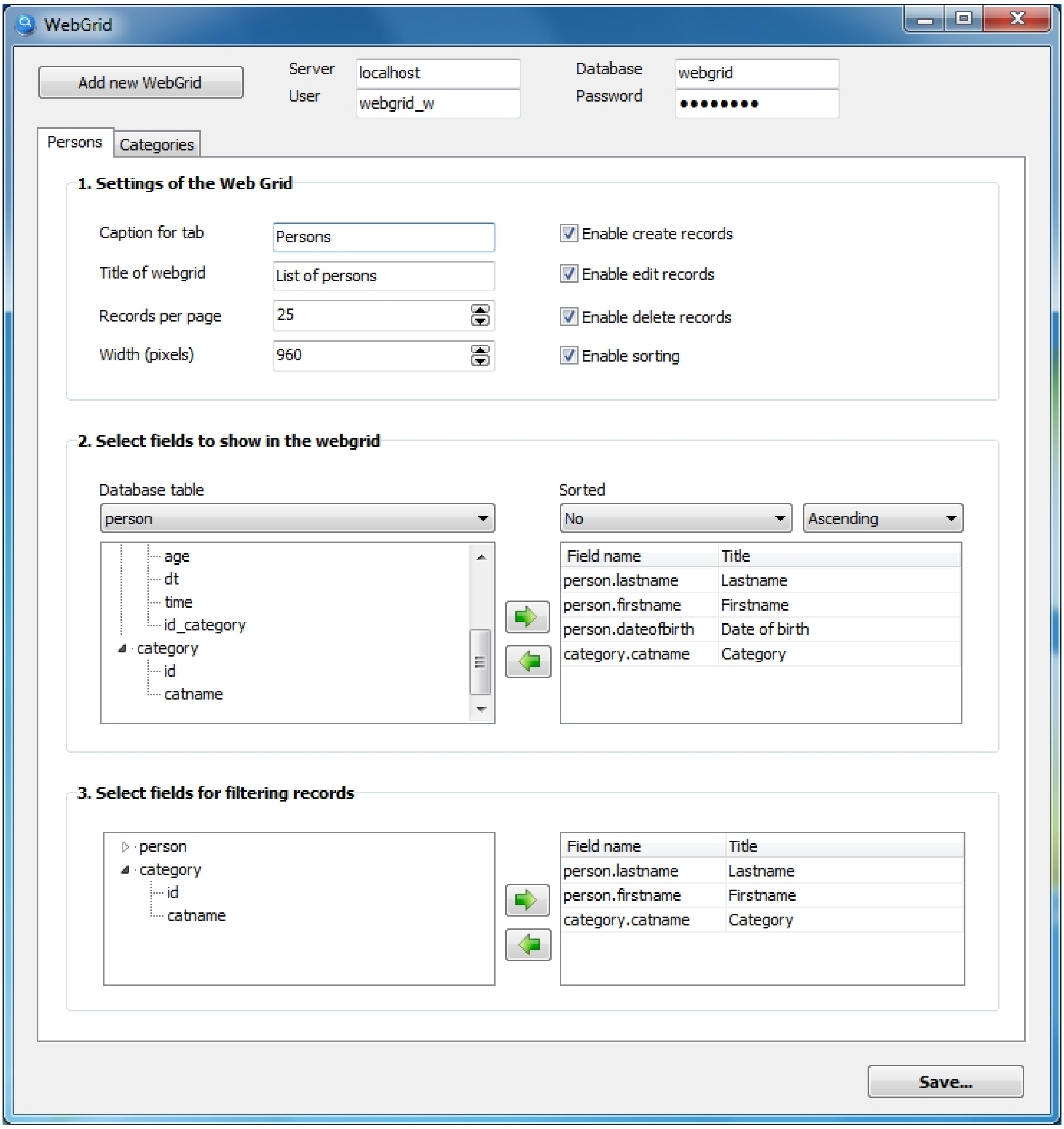
To use this functionality, your project must use DBMS MySQL. Read more about this here.

In addition, you need the simplest Web hosting, for example: https://www.hetzner.com/webhosting

It is assumed that you have already created the necessary structure of the database, only after that you can start creating web access.

Go to the menu: **Project > WebGrid**

This tool allows you to create web access to any number of tables in your database. Click the "Add new WebGrid" button.
In the figure below you can see an example of how to set up a web table.



Server: localhost

User: webgrid_w

Database: webgrid

Password: ••••••••

1. Settings of the Web Grid

Caption for tab: Persons

Title of webgrid: List of persons

Records per page: 25

Width (pixels): 960

☑ Enable create records

☑ Enable edit records

☑ Enable delete records

☑ Enable sorting

2. Select fields to show in the webgrid

Database table: person

```
    age
    dt
    time
    id_category
◢ category
    id
    catname
```

Sorted: No / Ascending

Field name	Title
person.lastname	Lastname
person.firstname	Firstname
person.dateofbirth	Date of birth
category.catname	Category

3. Select fields for filtering records

```
▷ person
◢ category
    id
    catname
```

Field name	Title
person.lastname	Lastname
person.firstname	Firstname
category.catname	Category

You can click the "Add new WebGrid" button again to create the necessary number of web tables that will be available in the browser through tabs.

After setting up all the web tables you need, click the "**Save...**" button, then the program will ask you to specify the folder on your computer where will be saved the files (php+css+html) that you will need to upload to your web server.

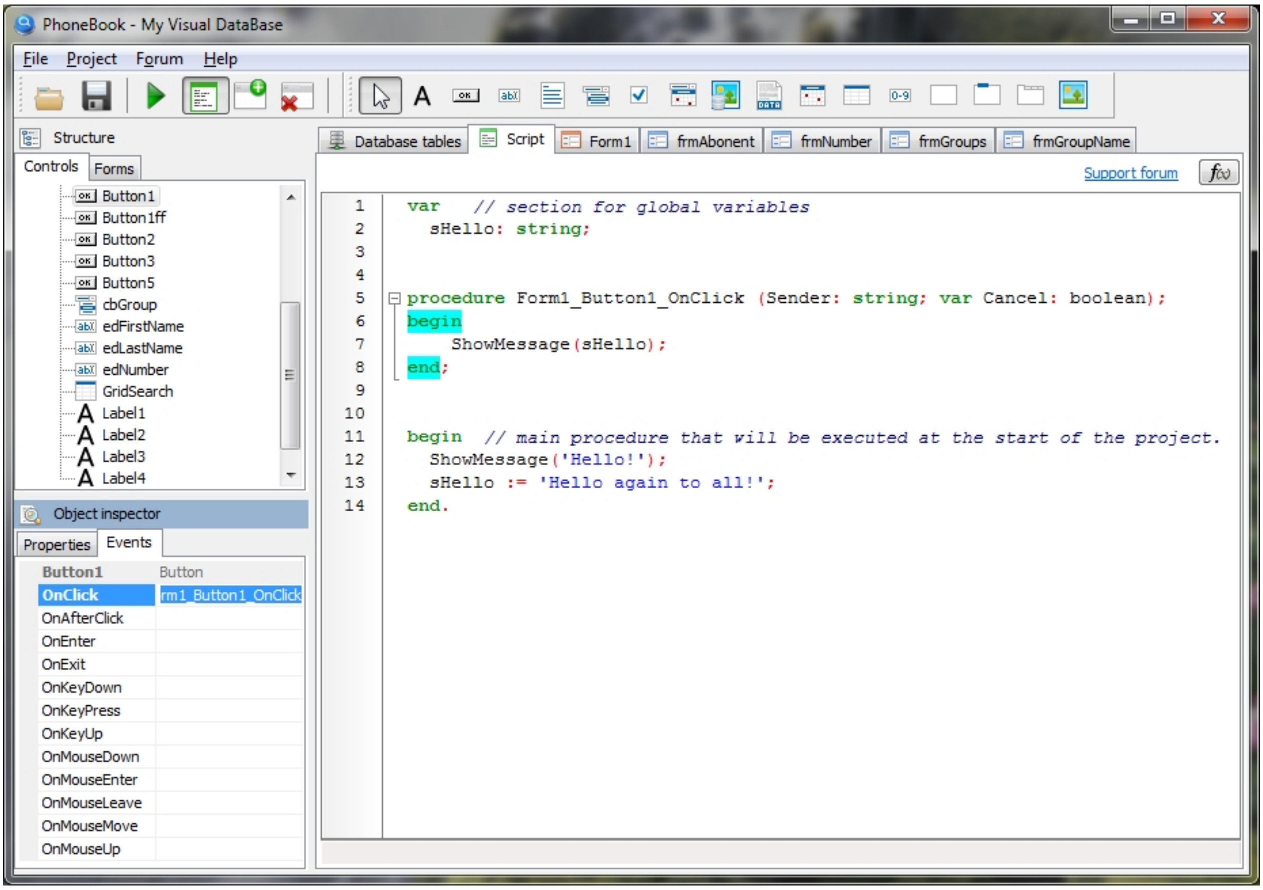
Script

Introduction

Scripts allow you to implement almost any application logic, interact with visual components and the database.
In addition, there are a large number of classes with different purposes, such as file management, graphics, timers, etc.

At the same time, you can create a full-fledged accounting program without using scripts.

In order to use scripts in your project, you just need to click on the button on the toolbar, after that the "Script" tab will appear, where you will write scripts. You will also see an additional "Events" tab in the "Object inspector" panel.



Button1	Button
OnClick	rm1_Button1_OnClick

```
var   // section for global variables
  sHello: string;


procedure Form1_Button1_OnClick (Sender: string; var Cancel: boolean);
begin
    ShowMessage(sHello);
end;


begin   // main procedure that will be executed at the start of the project.
  ShowMessage('Hello!');
  sHello := 'Hello again to all!';
end.
```

The figure shows an example of a simple script that shows a greeting message when the project starts, and also shows a greeting when the Button1 button is pressed.

Quite often you will need to use events from various components. Let's look at an example.

Note the event handler: **procedure Form1_Button1_OnClick (Sender: string; var Cancel: boolean);**

It was created as follows, go to the tab "Events", and double-click on the blank line, opposite the event you want, in our case OnClick.

The name of the procedure for the event will be generated automatically, and now between the keywords begin and end; you can write the necessary script that will be executed when this event occurs, for example, when the user clicks on this button.

Each visual component has many kinds of events with which you can implement the program behavior you want.

Pascal language

The script is a fairly popular programming language Object Pascal, which is used in the Delphi programming environment. You can easily find many self-study guides on the Internet for this programming language.

Below you will find useful links for learning:

http://www.marcocantu.com/epascal/
http://101.lv/learn/delphi/
http://www.delphibasics.co.uk/
http://delphi.about.com/od/beginners/a/dbeginner6.htm
http://www.delphibasics.co.uk/Article.asp?Name=FirstPgm
https://blog.udemy.com/pascal-programming/

Component Properties, Methods and Events

Form

Description

Form represents a standard application window (form).

Class: TAForm

Properties

Property	Type	Description
dbAction	String	Contains the name of the action of the button with which the form was calle empty value indicates that the form was opened with a script. Read-only pr
AlphaBlend	Boolean	Specifies whether the form is translucent. Works since Windows 2000
AlphaBlendValue	Integer	Specifies the degree of translucency on a translucent form. The value is fro
AutoScroll	Boolean	Indicates whether scroll bars appear automatically on the scrolling window
AutoSize	Boolean	Specifies whether the control sizes itself automatically to accommodate its
BorderIcons	TBorderIcons	Specifies which icons appear on the title bar of the form. More info.
BorderStyle	TBorderStyle	Specifies the appearance and behavior of the form border. More info.
Canvas	TCanvas	Provides access to the drawing area of the form. More info.
Caption	String	Window caption.
CalledForm	TAForm	Reference to the form from which the current form was called. Contains ar

ClientWidth	Integer	Specifies the horizontal size of the form's client area in pixels (without bor
ClientHeight	Integer	Specifies the height of the form's client area in pixels (without borders and
Color	TColor	Specifies the background color of the control. More info.
ComponentCount	Integer	Indicates the number of components owned by the component.
Components[i]	TComponent	Allows you to refer to a component on a form by its index.
ControlCount	Integer	Returns the number of child controls.
Controls[i]	TControl	Allows you to refer to a child component on a form by its index.
Constraints	TSizeConstraints	Specifies the size constraints for the control. More info.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes ir
Enabled	Boolean	Controls whether the control responds to mouse, keyboard, and timer eve
Font	TFont	Allows you to set font name, size, color and style. More info.
FormStyle	TFormStyle	Determines the form's style. Values: fsNormal, fsMDIChild, fsMDIForm, fsSt
HorzScrollBar	TControlScrollBar	Configuring the appearance and behavior of horizontal scrolling, see also
KeyPreview	Boolean	Specifies whether the form should receive keyboard events before the acti
Name	String	The name of the form.
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
TransparentColor	Boolean	Specifies whether a color on the form appears transparent.
TransparentColorValue	TColor	Indicates the color on the form that appears transparent when Transparen
VertScrollBar	TControlScrollBar	Configuring the appearance and behavior of the vertical scrolling, see also
Visible	Boolean	Specifies whether the form appears onscreen.
WindowState	TWindowState	Represents how the form appears on the screen. Values: wsNormal, wsMi
Left	Integer	Specifies the horizontal coordinate of the left edge of a form relative to th
Top	Integer	Specifies the Y coordinate of the upper-left corner of a form, relative to th
Width	Integer	Specifies the horizontal size of the form in pixels.
Height	Integer	Specifies the vertical size of the form in pixels.

Methods

Method	**Description**
function **CanFocus**: Boolean	Indicates whether a control can receive
procedure **Close**	Closes the form.
function **FindComponent** (const AName: string): TComponent	Indicates whether a given component is
procedure **Hide**	Hides the form.
procedure **NewRecord** (ParentTable: string = ''; ParentTableID: integer = -1)	Prepares and shows the form on the scr
procedure **ScaleBy** (M, D: Integer)	Rescale control and its children.
procedure **SetFocus**	Sets focus to the form.
procedure **SetFocusNextControl**	Passes the input focus to the next comp
procedure **Show**	Shows the form.
procedure **ShowModal**	Use ShowModal to show a form as a m return until the form closes.
procedure **ShowRecord** (TableName: string; id: integer)	Displays a form with data from the data

Events

Event	**Description**

OnClick	Occurs when the user clicks the control.
OnClose	Occurs when the form closes.
OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the fo
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a form.
OnMouseEnter	Occurs when the user moves the mouse into a form.
OnMouseLeave	Occurs when the user moves the mouse outside of a form.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a form.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a
OnResize	Occurs immediately after the form is resized.
OnDropFiles	Occurs when user tries to drag and drop a file from explorer to a form. More info.
OnShow	Occurs when the form is shown (that is, when its Visible property is set to true).

BorderIcons

Type

TBorderIcons

Description

Specifies which icons appear on the title bar of the form.

biSystemMenu - the form has a Control menu (also known as a System menu)
biMinimize - the form has a Minimize button
biMaximize - the form has a Maximize button
biHelp - If BorderStyle is bsDialog or biMinimize and biMaximize are excluded, a question mark appears in the form's title bar and when clicked, the cursor changes to crHelp; otherwise,no question mark appears.

These values can be combined with the + sign

Example

```
Form1.BorderIcons:= biSystemMenu + biMinimize;
Form1.BorderIcons:= biSystemMenu + biMaximize;
Form1.BorderIcons:= 0; // allows you to hide all the system window buttons
```

BorderStyle

Type

TBorderStyle

Description

Specifies the appearance and behavior of the form border.

The following values are available:

bsDialog - not resizable; no minimize/maximize menu
bsNone - not resizable; no visible border line
bsSingle - not resizable; minimize/maximize menu
bsSizeable - standard resizable border
bsSizeToolWin - like bsSizeable with a smaller caption
bsToolWindow - like bsSingle but with a smaller caption

Example

```
Form1.BorderStyle := bsDialog;
```

TControlScrollBar

Description

The class allows you to set parameters for scrollbars on a form.

The class has the following properties

Property	Description
IsScrollBarVisible: Boolean	Returns true if the scroll bar is visible.
ScrollPos: Integer	Indicates the position of the thumb tab.
ButtonSize: Integer	Specifies the sizes of the buttons in the scroll bar.
Position: Integer	Specifies the position of the thumb tab on the scroll bar.
Range: Integer	Determines how far the scrolling region of the associated control can move.
Tracking: Boolean	Determines whether the form or scroll box moves before the thumb tab is released.
Visible: Boolean	Determines whether the scroll bar appears.

Exampple

```
Form1.HorzScrollBar.Position := 100;
Form1.VertScrollBar.Tracking := True;
```

Label

Description

Use Label to add text that the user cannot edit on a form. This text can be used to label another control.

Class: TdbLabel

Properties

Property	Type	Description
AutoSize	Boolean	Determines whether the size of the label automatically resizes to accommo
Caption	String	Specifies a text string that identifies the control to the user.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes in
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
Font	TFont	Allows you to set font name, size, color and style. More info.
Hint	String	Hint contains the text string that appears when the user moves the mouse
Name	String	The name of the component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves ov
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Visible	Boolean	Specifies whether the component appears onscreen.
WordWrap	Boolean	Specifies whether the label text wraps when it is too long for the width of t
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Events

Event	Description
OnClick	Occurs when the user clicks the component.
OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the co
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a component.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a component.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a

Button

Description

The component is used to perform actions when the user clicks on it.

Class: TdbButton

Properties

Property	Type	Description
dbGeneralTableId	Integer	Makes sense if the button has the "Save record" action assigned to it. Con
dbGeneralTable	String	It makes sense if the button has a "Search", "Save Record" or "Report" acti
dbGetSqlStatement	String	Makes sense if the button has a "Search" or "Report" action. Contains the
dbParentTableId	Integer	Makes sense if the button has a "Save record" action. Contains the id of th
dbParentTable	Integer	Makes sense if the button has a "Save record" action. Contains the name c
dbSQL	String	It makes sense if the button has an "SQL query" or "Report (SQL)" action a
dbFilter	String	Makes sense if the button has a "Search" or "Report" action. Allows you to
dbReportFile	String	Makes sense if the button has a Report or Report (SQL) action assigned tc that the file is located in the Report folder of your project.
dbReportResultFile	String	Makes sense if the button has a Report or Report (SQL) action assigned tc
dbReportOpenIn	TReportOpenIn	Makes sense if the button has a Report or Report (SQL) action assigned tc rpoPrintQuick, rpoExcel, rpoWord, rpoPDF, rpoHTML, rpoODT, rpoODS, rp
dbActionType	TActionDbType	Defines the action of the button. Available values: adbNone, adbSearch, ac adbShowForm, adbCloseForm, adbGridToExcel
dbDoCloseForm	Boolean	Makes sense if the button has a "Save Record" action assigned to it. Deter
dbDontResetID	Boolean	Makes sense if the button has a "Save Record" action assigned to it. Deter
Cancel	Boolean	Determines whether the button is automatically pressed when the user on
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary b if the component is located on a parent component with those properties, example: if Form1.Button1.CanFocus then Form1.Button1.SetFocus;
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes ir
Caption	String	Specifies a text string that identifies the control to the user.
Default	Boolean	Determines whether the button is automatically pressed when the Enter key
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
Hint	String	Hint contains the text string that appears when the user moves the mouse
ImageMargins	TImageMargins	Margins of image on button. Example: Form1.Button1.ImageMargins.Left :=
Name	String	The name of the component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves ov
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrde
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allov
Visible	Boolean	Specifies whether the component appears onscreen.
WordWrap	Boolean	Specifies whether the button text wraps to fit the width of the control.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative

Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	Description
procedure **Click**	Allows you to click the button, thereby performing the action specified in the Action (dbActionTyp
procedure **SetFocus**	Gives the input focus to the component.

Events

Event	Description
OnClick	Occurs when the user clicks the component. The event also allows you to prevent the selected a
OnAfterClick	Occurs when you click on the component after the action specified for the button. If no action is
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a component.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a component.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a
OnDropFiles	Occurs when user tries to drag and drop a file from explorer to a form. More info.

OnClick

Description

Occurs when the user clicks the component. The event also allows you to prevent the selected action for the button.

Examples

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
begin
    Cancel := True; // If the Cancel parameter is set to True, the action that
is assigned to the button will not be executed
end;
```

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
begin
    if IDNO = MessageBox('Execute an action?', 'Caption',
MB_YESNO+MB_ICONQUESTION) then // If the user clicked No
    begin
        Cancel := True; // If the Cancel parameter is set to True, the action
that is assigned to the button will not be executed
    end;
end;
```

Edit

Description

The component is used to input and output text/numeric information.

Class: TdbEdit

Properties

Property	Type	Description
sqlValue	String	Returns the value of a component, for use in SQL queries. The property va Currency = True, the escape quotes will be omitted. In case of an empty va *example:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+Forr
Alignment	TAlignment	Determines how the text is aligned within the text edit control. Available va
AutoSelect	Boolean	Determines whether all the text in the edit control is automatically selected
BorderStyle	TBorderStyle	Determines whether the edit control has a single line border around the cl
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary False property, or if the component is located on a parent component with example: if Form1.Edit1.CanFocus then Form1.Edit1.SetFocus;
CharCase	TEditCharCase	Determines the case of the text within the edit control. Available values: ec
Color	TColor	Specifies the background color of the control. More info.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes ir
dbFilter	String	It makes sense when the component is used together with the button with
dbTable	String	Determines which database table a component belongs to.
dbField	String	Determines which field of the database table this component belongs to.
dbIncremSearch	String	Allows you to specify the name of the button on the current form with the text for instant search.
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
GetTextLen	Integer	Returns the length of the control's text.

Hint	String	Hint contains the text string that appears when the user moves the mouse
MaxLength	Integer	Specifies the maximum number of characters the user can enter into the ed
Name	String	The name of the component.
NumbersOnly	Boolean	Allows only numbers to be typed into the text edit.
PasswordChar	String	Indicates the character, if any, to display in place of the actual characters t used: *
ReadOnly	Boolean	Determines whether the user can change the text of the edit component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves o
SelLength	Integer	Specifies the number of characters that are selected.
SelStart	Integer	Specifies the position of the first selected character in the text. If there is n
SelText	String	Specifies the selected portion of the edit component's text.
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrde
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allo
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Text	String	Contains a text string associated with the component.
TextHint	String	A hint or message to be displayed when the Text property is empty.
Value	Double	The numerical value of the component.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	**Description**
procedure **Clear**	Deletes all text from the edit component.
procedure **CopyToClipboard**	Copies the selected text in the edit component to the Clipboard.
procedure **CutToClipboard**	Copies the selected text to the Clipboard and then deletes the selection.
procedure **PasteFromClipboard**	Pastes the contents of the Clipboard into edit component, replacing the current sel
procedure **SelectAll**	Selects all text in the edit component.
procedure **SetFocus**	Gives the input focus to the component.

Events

Event	**Description**
OnChange	Occurs when the text for the edit component may have changed.
OnClick	Occurs when the user clicks the component.
OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the c
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a component.

OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a component.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a
OnDropFiles	Occurs when user tries to drag and drop a file from explorer to a form. More info.

Memo

Description

The component is used to input and output multiline text information.

Class: TdbMemo

Properties

Property	Type	Description
sqlValue	String	Returns the value of a component, for use in SQL queries. The property va *example:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+Forı
Alignment	TAlignment	Determines how the text is aligned within the text edit control. Available va
AutoSelect	Boolean	Determines whether all the text in the component is automatically selected
BorderStyle	TBorderStyle	Determines whether the edit control has a single line border around the cl
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary False property, or if the component is located on a parent component with example: if Form1.Memo1.CanFocus then Form1.Memo1.SetFocus;
CaretPosX	Integer	Indicates the X position of the caret in the client area of the memo.
CaretPosY	Integer	Indicates the Y position of the caret in the client area of the memo.
CharCase	TEditCharCase	Determines the case of the text within the edit control. Available values: ecl
Color	TColor	Specifies the background color of the control. More info.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes ir
dbTable	String	Determines which database table a component belongs to.
dbField	String	Determines which field of the database table this component belongs to.
dbIncremSearch	String	Allows you to specify the name of the button on the current form with the text for instant search.
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
GetTextLen	Integer	Returns the length of the component's text.
Hint	String	Hint contains the text string that appears when the user moves the mouse
Lines	TStringList	Class that represent a list of strings. More info.
Lines[i]	String	Contains the individual lines of text in the memo component. Examle Show
MaxLength	Integer	Specifies the maximum number of characters the user can enter into the m
Name	String	The name of the component.

ReadOnly	Boolean	Determines whether the user can change the text of the memo component
ScrollBars	TScrollStyle	Determines whether the memo control has scroll bars. Available values: ss
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves ov
SelLength	Integer	Specifies the number of characters that are selected.
SelStart	Integer	Specifies the position of the first selected character in the text. If there is n
SelText	String	Specifies the selected portion of the memo component's text.
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrde
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allo
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Text	String	Contains a text string associated with the component.
TextHint	String	A hint or message to be displayed when the Text property is empty.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	**Description**
procedure **Clear**	Deletes all text from the memo component.
procedure **CopyToClipboard**	Copies the selected text in the memo component to the Clipboa
procedure **CutToClipboard**	Copies the selected text to the Clipboard and then deletes the s
procedure **PasteFromClipboard**	Pastes the contents of the Clipboard into memo component, rep
procedure **SaveToFileUTF8** (const FileName: string)	Saves the content of the component in a UTF-8 encoded text file
procedure **SelectAll**	Selects all text in the memo component.
procedure **SetFocus**	Gives the input focus to the component.

Events

Event	**Description**
OnClick	Occurs when the user clicks the component.
OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the c
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a component.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a component.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a
OnDropFiles	Occurs when user tries to drag and drop a file from explorer to a form. More info.

RichEdit

Description

The component is an advanced text editor with wide possibilities for text formatting. With the ability to insert graphics files, tables, links, etc. The data storage format is RTF (Rich Text Format).

Class: TdbRichEdit

Properties

Property	Type	Description
dbTable	String	Determines which database table a component belongs to.
dbField	String	Determines which field of the database table this component belongs to.
sqlValue	String	Returns the value of a component, for use in SQL queries. The property va *example:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+Forr
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
ItemCount	Integer	Number of items in the component.
Name	String	The name of the component.
Ruler	Boolean	Determines the visibility of the rulers.
SelLength	Integer	Specifies the number of characters that are selected.
SelStart	Integer	Specifies the position of the first selected character in the text. If there is n
Modified	Boolean	Value of this property is True if document was modified
ReadOnly	Boolean	Determines whether the user can change the text of the RichEdit compone
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrde
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allov
Text	String	Contains a text string associated with the component.
TextRTF	String	Allows you to retrieve and assign text to a document in RTF format.
ToolBar1	Boolean	Defines the visibility of the first row toolbar.
ToolBar2	Boolean	Defines the visibility of the second row toolbar.
ToolBar3	Boolean	Defines the visibility of the third row toolbar.
Visible	Boolean	Specifies whether the component appears onscreen.
VScrollVisible	Boolean	Set to False to hide vertical scrollbar.
WheelStep	Integer	Defines how much the document will be scrolled when the user turns a mo
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.

Methods

Method
procedure **AddHotPicture** (const Name: String; gr: TGraphic; ParaNo: Integer = -1; VAlign: TRVVAlign = rvvaBaseline)
procedure **AddHyperlink** (const s: String; url: String)
procedure **AddNL** (const s: String; StyleNo: Integer; ParaNo: Integer = -1)
procedure **AddPicture** (const Name: String; gr: TGraphic; ParaNo: Integer = -1; VAlign: TRVVAlign = rvvaBaseline)
procedure **AddTab** (TextStyleNo, ParaNo: Integer)
procedure **AddTextNL** (const s: String; StyleNo, FirstParaNo, OtherParaNo: Integer; AsSingleParagraph: Boolean = False)
function **AppendRTF** (const FileName: String): Boolean
function **AppendRTFFromStream** (Stream: TStream): Boolean
function **AppendTextA** (const FileName: String; StyleNo, ParaNo: Integer; AsSingleParagraph: Boolean): Boolean
function **AppendText** (const FileName: String; StyleNo, ParaNo: Integer; DefAsSingleParagraph: Boolean): Boolean
function **AppendTextFromStreamA** (Stream: TStream; StyleNo, ParaNo: Integer; AsSingleParagraph: Boolean): Boolean
function **AppendTextFromStream** (Stream: TStream; StyleNo, ParaNo: Integer; DefAsSingleParagraph: Boolean): Boolean
procedure **ApplyTextStyle** (TextStyleNo: Integer)
function **CanFocus**: Boolean
procedure **Clear**
procedure **Copy**
procedure **CopyImage**
procedure **CopyRTF**
procedure **CopyText**
procedure **CopyTextA**
procedure **DeleteItems** (FirstItemNo, Count: Integer)
procedure **DeleteSelection**
procedure **Deselect**
function **Focused**: Boolean
procedure **Format**
procedure **FormatTail**
procedure **Reformat**
function **GetSelectedImage**: TGraphic
function **GetSelText**: String
procedure **InsertHyperlink** (const s: String; url: string)
function **InsertPicture** (const Name: String; gr: TGraphic; VAlign: TRVVAlign = rvvaBaseline): Boolean
function **InsertRTFFromFileEd** (const FileName: String): Boolean
function **InsertRTFFromStreamEd** (Stream: TStream): Boolean
procedure **InsertText** (const text: String; CaretBefore: Boolean=False)
procedure **InsertTextEx** (const text: String; FontColor: TColor = -1; FontSize: integer = -1; FontStyles: Integer = -1; FontNa
function **InsertTextFromFileA** (const FileName: String): Boolean
function **InsertTextFromFile** (const FileName: String): Boolean
function **LoadHTML** (const FileName: String): Boolean
function **LoadRTF** (const FileName: String): Boolean
function **LoadRTFFromStream** (Stream: TStream): Boolean

function **LoadTextA** (const FileName: String; StyleNo, ParaNo: Integer; AsSingleParagraph: Boolean): Boolean
function **LoadText** (const FileName: String; StyleNo, ParaNo: Integer; DefAsSingleParagraph: Boolean): Boolean
function **LoadTextFromStreamA** (Stream: TStream; StyleNo, ParaNo: Integer; AsSingleParagraph: Boolean): Boolean
function **LoadTextFromStream** (Stream: TStream; StyleNo, ParaNo: Integer; DefAsSingleParagraph: Boolean): Boolean
procedure **Paste**
procedure **PasteRTF**
procedure **PasteText**
procedure **PasteTextA**
function **SaveDocX** (const FileName: String; SelectionOnly: Boolean = False): Boolean
function **SaveDocXToStream** (Stream: TStream; SelectionOnly: Boolean = False): Boolean
function **SaveHTML** (FileName, Title: String; ImagesPrefix: String = ''): Boolean
function **SaveHTMLEx** (const FileName, Title, ImagesPrefix, ExtraStyles, ExternalCSS: String): Boolean
function **SaveHTMLToStream** (Stream: TStream; const Path, Title, ImagesPrefix: String): Boolean
function **SaveRTF** (const FileName: String; SelectionOnly: Boolean = False): Boolean
function **SaveRTFToStream** (Stream: TStream; SelectionOnly: Boolean = False): Boolean
function **SaveText** (const FileName: String; LineWidth: Integer = 0): Boolean
function **SaveTextA** (const FileName: String; LineWidth: Integer = 0): Boolean
function **SaveTextToStreamA** (const Path: String; Stream: TStream; LineWidth: Integer; SelectionOnly, TextOnly: Boolean):
function **SaveTextToStream** (const Path: String; Stream: TStream; LineWidth: Integer; SelectionOnly, TextOnly: Boolean): B
function **SearchText** (s: String; MatchCase: boolean = False; Down: boolean = True; WholeWord: boolean = False; MultiIt True; SmartStart: boolean = False): Boolean
procedure **SelectAll**
procedure **SetFocus**
procedure **SelectionToHyperlink** (url: string)

Events

Event	Description
OnClick	Occurs when the user clicks the component.
OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the c
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a component.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a component.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a

OnDropFiles	Occurs when user tries to drag and drop a file from explorer to a form. More info.

AddHotPicture

Description

Adds picture-hyperlink to the end of document.

```
procedure AddHotPicture (const Name: String; gr: TGraphic; ParaNo: Integer = -
1; VAlign: TRVVAlign = rvvaBaseline);
```

Parameter	Description
Name	Name of this hot-picture item, any string. Name must not contain CR and LF characters. RichEidt does no
gr	Picture to insert.
ParaNo	Optional parameter. If ParaNo=-1, the method adds an item to the end of the last paragraph. If ParaNo>
VAlign	Optional parameter. Vertical align of this picture, relative to its line. Available values: rvvaBaseline, rvvaMid

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
   Graphic: TGraphic;
begin
   Graphic := TJpegImage.Create;
   Graphic.LoadFromFile('d:\filename.jpg');
   Form1.RichEdit1.AddHotPicture('', Graphic);
   Form1.RichEdit1.Format;
end;
```

AddHyperlink

Description

Adds a link to the end of the document.

```
procedure AddHyperlink (const s: String; url: String);
```

Parameter	Description
s	Текст, который будет ссылкой.
url	Ссылка на web сайт либо локальный файл.

Example

```
   Form1.RichEdit1.AddHyperlink('текст ссылки', 'http://google.com');
   Form1.RichEdit1.AddHyperlink('текст ссылки', 'd:\picture.jpg');
   Form1.RichEdit1.Format; // применям внесенные изменения в документ
```

AddNL

Description

Adds a text element to the end of the document.

```
procedure AddNL(const s: String; StyleNo: Integer; ParaNo: Integer = -1);
```

Parameter	Description
s	Text string to add. It must not contain CR, LF, TAB, FF characters (#13, #10, #9, #12). To add several lines o
StyleNo	Style number. Not used, use the value 0.
ParaNo	Optional parameter. If ParaNo=-1, the methods add an item to the end of the last paragraph. If ParaNo>=0,

Example

```
   Form1.RichEdit1.AddNL('Hello', 0);
   Form1.RichEdit1.Format; // To apply the changes to the document
```

AddPicture

Description

Adds a picture to the end of document.

```
procedure AddPicture (const Name: String; gr: TGraphic; ParaNo: Integer = -1;
VAlign: TRVVAlign = rvvaBaseline);
```

Parameter	Description
Name	Name of this picture item, any string. Name must not contain CR and LF characters. RichEdit does not use
gr	Picture to insert. By default, this picture will be owned by RichEdit component, and you must not free it.
ParaNo	Optional parameter. If ParaNo=-1, the method adds an item to the end of the last paragraph. If ParaNo>
VAlign	Optional parameter. Vertical align of this picture, relative to its line, available values: rvvaBaseline, rvvaMic

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
   Graphic: TGraphic;
begin
   Graphic := TJpegImage.Create;
   Graphic.LoadFromFile('d:\filename.jpg');
```

```
    Form1.RichEdit1.AddPicture('', Graphic);
    Form1.RichEdit1.Format;
end;
```

AddTab

Description

Adds tabulator to the end of document.

```
procedure AddTab (TextStyleNo, ParaNo: Integer);
```

Parameter	Description
TextStyleNo	The number of the text style. Not used, use the value 0.
ParaNo	If ParaNo=-1, the method adds an item to the end of the last paragraph. If ParaNo>=0, this item starts

Example

```
    Form1.RichEdit1.AddTab(0, -1);
    Form1.RichEdit1.Format; // To apply the changes to the document
```

AddTextNL

Description

Adds one or more text lines to the end of document.

```
procedure AddTextNL (const s: String; StyleNo, FirstParaNo, OtherParaNo:
Integer; AsSingleParagraph: Boolean = False)
```

Parameter	Description
s	Text string to add. It may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12).
StyleNo	The number of the text style. Not used, use the value 0.
FirstParaNo	If FirstParaNo=-1, the method adds an item to the end of the last paragraph. If FirstParaNo>=0, t
OtherParaNo	Defines paragraph attributes for the subsequent lines of text. Must be >=0.
AsSingleParagraph	If False, each new line will be added as a new paragraph. If True, it treats #13 and #10 as line bre

Example

```
    Form1.RichEdit1.AddTextNL('текст', 0, -1, 0);
    Form1.RichEdit1.Format; // To apply the changes to the document
```

AppendRTFFromStream

Description

Adding the contents of an RTF stream to a document.

```
function AppendRTFFromStream (Stream: TStream): Boolean
```

Parameter	Description
Stream	The stream that contains the RTF document.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    RtfFile: TFileStream;
begin
    RtfFile := TFileStream.Create('d:\document.rtf', fmOpenRead);
    try
        Form1.RichEdit1.AppendRTFFromStream(RtfFile);
    finally
        RtfFile.Free;
    end;
end;
```

AppendTextA

Description

Adding an ANSI encoded text file to the document.

```
function AppendTextA (const FileName: String; StyleNo, ParaNo: Integer;
AsSingleParagraph: Boolean): Boolean;
```

Parameter	Description
FileName	A text file in ANSI encoding.
StyleNo	The number of the text style. Not used, use the value 0.
ParaNo	If ParaNo=-1, the method adds an item to the end of the last paragraph. If ParaNo>=0, this item
AsSingleParagraph	If False, each new line will be added as a new paragraph. If True, it treats #13 and #10 as line br

Example

```
Form1.RichEdit1.AppendTextA('d:\file.txt', 0, -1, True);
```

AppendText

Description

Adding an UTF-16 encoded text file to the document.

```
function AppendText (const FileName: String; StyleNo, ParaNo: Integer;
DefAsSingleParagraph: Boolean): Boolean
```

Parameter	Description
FileName	A text file in UTF-16 encoding.
StyleNo	The number of the text style. Not used, use the value 0.
ParaNo	Optional parameter. If ParaNo=-1, the method adds an item to the end of the last paragraph.
DefAsSingleParagraph	If False, each new line will be added as a new paragraph. If True, it treats #13 and #10 as line

Example

```
Form1.RichEdit1.AppendText('d:\file.txt', 0, -1, True);
```

AppendTextFromStreamA

Description

Adding ANSI encoded text stream content to the document.

```
function AppendTextFromStreamA (Stream: TStream; StyleNo, ParaNo: Integer;
AsSingleParagraph: Boolean): Boolean;
```

Parameter	Description
Stream	The stream that contains the ANSI encoded text file.
StyleNo	The number of the text style. Not used, use the value 0.
ParaNo	Optional parameter. If ParaNo=-1, the method adds an item to the end of the last paragraph. If P
AsSingleParagraph	If False, each new line will be added as a new paragraph. If True, it treats #13 and #10 as line bre

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    TxtFile: TFileStream;
begin
    TxtFile := TFileStream.Create('d:\file.txt', fmOpenRead);
    try
        Form1.RichEdit1.AppendTextFromStreamA(TxtFile, 0, -1, True);
    finally
```

```
        TxtFile.Free;
    end;
end;
```

AppendTextFromStream

Description

Adding the contents of a UTF-16 encoded text stream to the document.

```
function AppendTextFromStream (Stream: TStream; StyleNo, ParaNo: Integer;
AsSingleParagraph: Boolean): Boolean;
```

Parameter	Description
Stream	The stream that contains the UTF-16 encoded text file.
StyleNo	The number of the text style. Not used, use the value 0.
ParaNo	Optional parameter. If ParaNo=-1, the method adds an item to the end of the last paragraph. If
AsSingleParagraph	If False, each new line will be added as a new paragraph. If True, it treats #13 and #10 as line br

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    TxtFile: TFileStream;
begin
    TxtFile := TFileStream.Create('d:\file.txt', fmOpenRead);
    try
        Form1.RichEdit1.AppendTextFromStream(TxtFile, 0, -1, True);
    finally
        TxtFile.Free;
    end;
end;
```

GetSelectedImage

Description

This method returns image, if the selection consist only of one image. If there is nothing selected, or not only image is selected, this method returns nil. The method returns image owned by RichView, not a copy of it. So do not destroy this image.

This method must be called only when the document is formatted. To format it, call Format method.

```
function GetSelectedImage: TGraphic;
```

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    g: TGraphic;
begin
    g := Form1.RichEdit1.GetSelectedImage;
    if g <> nil then Form1.Image1.Picture.Assign(g);
end;
```

InsertHyperlink

Description

Inserts a link at the cursor position.

```
procedure InsertHyperlink (const s: String; url: String);
```

Parameter	Description
s	The text that will be the link.
url	Link to a web site or local file.

Example

```
   Form1.RichEdit1.InsertHyperlink('link text', 'http://google.com');
   Form1.RichEdit1.InsertHyperlink('link text', 'd:\picture.jpg');
```

InsertPicture

Description

Inserts a picture at the cursor position.

```
function InsertPicture (const Name: String; gr: TGraphic; VAlign: TRVVAlign =
rvvaBaseline): Boolean;
```

Parameter	Description
Name	Name of this picture item, any string. Name must not contain CR and LF characters. RichEidt does not us
gr	Picture to insert.
VAlign	Optional parameter. Vertical align of this picture, relative to its line. Available values: rvvaBaseline, rvvaMid

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
   Graphic: TGraphic;
```

```
begin
   Graphic := TJpegImage.Create;
   Graphic.LoadFromFile('d:\filename.jpg');
   Form1.RichEdit1.InsertPicture('', Graphic);
end;
```

InsertRTFFromStreamEd

Description

Inserts the contents of the RTF stream into the document at the cursor position.

```
function InsertRTFFromStreamEd (Stream: TStream): Boolean
```

Parameter	Description
Stream	The stream that contains the RTF document.

Example

```
   procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
   var
       RtfFile: TFileStream;
   begin
       RtfFile := TFileStream.Create('d:\document.rtf', fmOpenRead);
       try
           Form1.RichEdit1.InsertRTFFromStreamEd(RtfFile);
       finally
           RtfFile.Free;
       end;
   end;
```

InsertText

Description

Inserts text at the cursor position.

```
procedure InsertText (const text: String; CaretBefore: Boolean=False);
```

Parameter	Description
text	Text may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12).
CaretBefore	Optional parameter. If CaretBefore=True, the caret will be positioned before the inserted text after inse

Example

```
   Form1.RichEdit1.InsertText('text');
   Form1.RichEdit1.InsertText('texts', True);
```

InsertTextEx

Description

Inserts text at the cursor position with the ability to specify color, size, style and font name.

```
procedure InsertTextEx (const text: String; FontColor: TColor = -1; FontSize:
integer = -1; FontStyles: Integer = -1; FontName: string = '');
```

Parameter	Description
text	Text may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12).
FontColor	Optional parameter. Font color. More about type TColor.
FontSize	Optional parameter. Font size.
FontStyles	Optional parameter. Font style. Available values: fsBold, fsItalic, fsUnderline, fsStrikeout
FontName	Optional parameter. Font name.

Example

```
   Form1.RichEdit1.InsertTextEx('text', clRed, 16,
fsBold+fsItalic+fsUnderline+fsStrikeout, 'Arial');
   Form1.RichEdit1.InsertTextEx('text', clGreen, 14, fsBold+fsItalic);
```

LoadRTFFromStream

Description

Appends the content of RTF (Rich Text Format) stream Stream to the document.

```
function LoadRTFFromStream (Stream: TStream): Boolean
```

Parameter	Description
Stream	The stream that contains the RTF document.

Example

```
   procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
   var
       RtfFile: TFileStream;
   begin
       RtfFile := TFileStream.Create('d:\document.rtf', fmOpenRead);
       try
           Form1.RichEdit1.LoadRTFFromStream(RtfFile);
       finally
           RtfFile.Free;
```

```
        end;
    end;
```

LoadTextA

Description

Append the content of the text file FileName in ANSI encoding to the document.

```
function LoadTextA (const FileName: String; StyleNo, ParaNo: Integer;
AsSingleParagraph: Boolean): Boolean;
```

Parameter	Description
FileName	A text file in ANSI encoding.
StyleNo	The number of the text style. Not used, use the value 0.
ParaNo	If ParaNo=-1, the method adds an item to the end of the last paragraph. If ParaNo>=0, this item
AsSingleParagraph	If False, each new line will be added as a new paragraph. If True, it treats #13 and #10 as line br

Example

```
    Form1.RichEdit1.LoadTextA('d:\file.txt', 0, -1, True);
```

LoadText

Description

Append the content of the text file FileName in UTF-16 encoding to the document.

```
function LoadText (const FileName: String; StyleNo, ParaNo: Integer;
DefAsSingleParagraph: Boolean): Boolean;
```

Parameter	Description
FileName	A text file in UTF-16 encoding.
StyleNo	The number of the text style. Not used, use the value 0.
ParaNo	If ParaNo=-1, the method adds an item to the end of the last paragraph. If ParaNo>=0, this ite
DefAsSingleParagraph	If False, each new line will be added as a new paragraph. If True, it treats #13 and #10 as line

Example

```
    Form1.RichEdit1.LoadText('d:\file.txt', 0, -1, True);
```

LoadTextFromStreamA

Description

Append ANSI encoded text stream to the document.

```
function LoadTextFromStreamA (Stream: TStream; StyleNo, ParaNo: Integer;
AsSingleParagraph: Boolean): Boolean;
```

Parameter	Description
Stream	The stream that contains the ANSI encoded text file.
StyleNo	The number of the text style. Not used, use the value 0.
ParaNo	Optional parameter. If ParaNo=-1, the method adds an item to the end of the last paragraph. If
AsSingleParagraph	If False, each new line will be added as a new paragraph. If True, it treats #13 and #10 as line b

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    TxtFile: TFileStream;
begin
    TxtFile := TFileStream.Create('d:\file.txt', fmOpenRead);
    try
        Form1.RichEdit1.LoadTextFromStreamA(TxtFile, 0, -1, True);
    finally
        TxtFile.Free;
    end;
end;
```

LoadTextFromStream

Description

Append UTF-16 encoded text stream to the document.

```
function LoadTextFromStream (Stream: TStream; StyleNo, ParaNo: Integer;
DefAsSingleParagraph: Boolean): Boolean;
```

Parameter	Description
Stream	The stream that contains the UTF-16 encoded text file.
StyleNo	The number of the text style. Not used, use the value 0.
ParaNo	Optional parameter. If ParaNo=-1, the method adds an item to the end of the last paragrap
DefAsSingleParagraph	If False, each new line will be added as a new paragraph. If True, it treats #13 and #10 as li

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
```

```
var
    TxtFile: TFileStream;
begin
    TxtFile := TFileStream.Create('d:\file.txt', fmOpenRead);
    try
        Form1.RichEdit1.LoadTextFromStream(TxtFile, 0, -1, True);
    finally
        TxtFile.Free;
    end;
end;
```

SaveDocX

Description

Exports document (or the selected part, if SelectionOnly=True) to the file FileName as DocX (Microsoft Word format).

```
function SaveDocX (const FileName: String; SelectionOnly: Boolean = False):
Boolean;
```

Parameter	Description
FileName	The name of the output DocX file.
SelectionOnly	Optional parameter. If True, only selected part of the document is saved.

Example

```
    if Form1.RichEdit1.SaveDocX('d:\file.docx') then
        ShowMessage('File saved successfully')
    else
        ShowMessage('There was an error during the export.');
```

SaveDocXToStream

Description

Exports document (or the selected part, if SelectionOnly=True) to the stream Stream as DocX (Microsoft Word format).

```
function SaveDocXToStream (Stream: TStream; SelectionOnly: Boolean = False):
Boolean
```

Parameter	Description
Stream	The stream into which the document is saved in DocX format.
SelectionOnly	Optional parameter. If True, only selected part of the document is saved.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    DocX: TFileStream;
begin
    DocX := TFileStream.Create('d:\file.docx', fmCreate);
    try
        if Form1.RichEdit1.SaveDocXToStream(DocX) then
            ShowMessage('File saved successfully')
        else
            ShowMessage('There was an error during the export.');
    finally
        DocX.Free;
    end;
end;
```

SaveHTML

Description

Exports document to HTML or XHTML file, using HTML tags like <font>, <b>, <div>, etc and a set of images (in separate files).

```
function SaveHTML (FileName, Title: String; ImagesPrefix: String = ''):
Boolean;
```

Parameter	Description
FileName	The name of the output HTML file.
Title	The title of the output HTML file.
ImagesPrefix	Optional parameter. The first part of names of images that will be saved with HTML document.

Example

```
 if Form1.RichEdit1.SaveHTML('d:\file.html', Title) then
     ShowMessage('File saved successfully')
 else
     ShowMessage('There was an error during the export.');
```

SaveHTMLEx

Description

Exports document to HTML or XHTML file, using CSS (Cascading Style Sheets) and a set of images (in separate files).

```
function SaveHTMLEx (const FileName, Title, ImagesPrefix, ExtraStyles,
ExternalCSS: String): Boolean;
```

Parameter	Description
FileName	The name of the output HTML file.
Title	The title of the output HTML file.
ImagesPrefix	Optional parameter. The first part of names of images that will be saved with HTML document.
ExtraStyles	Strings that can contain additional entries of CSS (usually you need not to use it, set to '').
ExternalCSS	If this string is not empty, this method uses external CSS instead of saving CSS into HTML file).

Example

```
    if Form1.RichEdit1.SaveHTMLEx('d:\file.html', 'Title', '', '', '') then
        ShowMessage('File saved successfully')
    else
        ShowMessage('There was an error during the export.');
```

SaveRTF

Description

Exports document (or the selected part, if SelectionOnly=True) to the file FileName as RTF (Rich Text Format).

```
function SaveRTF (const FileName: String; SelectionOnly: Boolean = False):
Boolean;
```

Parameter	Description
FileName	The name of the output RTF file.
SelectionOnly	Optional parameter. If True, only selected part of the document is saved.

Example

```
    if Form1.RichEdit1.SaveRTF('d:\file.rtf') then
        ShowMessage('File saved successfully')
    else
        ShowMessage('There was an error during the export.');
```

SaveRTFToStream

Description

Exports document (or the selected part, if SelectionOnly=True) to the Stream as RTF (Rich Text Format).

```
function SaveRTFToStream (Stream: TStream; SelectionOnly: Boolean = False):
Boolean
```

Parameter	Description
Stream	The stream into which the document is saved in RTF format.
SelectionOnly	Optional parameter. If True, only selected part of the document is saved.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    RTF: TFileStream;
begin
    RTF := TFileStream.Create('d:\file.rtf', fmCreate);
    try
        if Form1.RichEdit1.SaveRTFToStream(RTF) then
            ShowMessage('File saved successfully')
        else
            ShowMessage('There was an error during the export.');
    finally
        RTF.Free;
    end;
end;
```

SaveTextToStreamA

Description

Exports document or selection as ANSI text.

```
function SaveTextToStreamA (const Path: String; Stream: TStream; LineWidth:
Integer; SelectionOnly, TextOnly: Boolean): Boolean
```

Parameter	Description
Path	Path for saving images and other non-text items. See TextOnly parameter.
Stream	The stream into which the document is saved in text format.
LineWidth	LineWidth is used for saving breaks (they are saved as LineWidth '-' characters)
SelectionOnly	If True, only selected part of the document is saved.
TextOnly	If True, non-text items are ignored when saving. If False, text representation of items is saved.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    Txt: TFileStream;
begin
    Txt := TFileStream.Create('d:\file.txt', fmCreate);
    try
        if Form1.RichEdit1.SaveTextToStreamA('', Txt, 1, False, True) then
            ShowMessage('File saved successfully')
        else
            ShowMessage('There was an error during the export.');
    finally
```

```
        Txt.Free;
    end;
end;
```

SaveTextToStream

Description

Exports document or selection as UTF-16 text.

```
function SaveTextToStream (const Path: String; Stream: TStream; LineWidth:
Integer; SelectionOnly, TextOnly: Boolean): Boolean
```

Parameter	Description
Path	Path for saving images and other non-text items. See TextOnly parameter.
Stream	The stream into which the document is saved in text format.
LineWidth	LineWidth is used for saving breaks (they are saved as LineWidth '-' characters)
SelectionOnly	If True, only selected part of the document is saved.
TextOnly	If True, non-text items are ignored when saving. If False, text representation of items is saved.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    Txt: TFileStream;
begin
    Txt := TFileStream.Create('d:\file.txt', fmCreate);
    try
        if Form1.RichEdit1.SaveTextToStream('', Txt, 1, False, True) then
            ShowMessage('File saved successfully')
        else
            ShowMessage('There was an error during the export.');
    finally
        Txt.Free;
    end;
end;
```

SearchText

Description

Search for the substring s in the document.

```
function SearchText (s: String; MatchCase: boolean = False; Down: boolean =
True; WholeWord: boolean = False; MultiItem: boolean = True; SmartStart:
boolean = False): Boolean
```

Parameter	Description

s	Search string.
MatchCase	Optional parameter. If True, a character case is taken into account when comparing strings.
Down	Optional parameter. If True, the search is performed to the end of the document. If False, it is performed
WholeWord	Optional parameter. If True, the searched string matches only whole words.
MultiItem	Optional parameter. If True, the search can match substrings of several text items. If not included, the t if the document contains the text Hello, the substring 'Hello' can be found only if this option is included,
SmartStart	Not used.

Example

```
if Form1.RichEdit1.SearchText('string') then
   ShowMessage('Found')
else
   ShowMessage('Not found');


if Form1.RichEdit1.SearchText('string', True) then // case-sensitive search
   ShowMessage('Found')
else
   ShowMessage('Not found');
```

CheckBox

Description

CheckBox represents a check box that can be on (checked) or off (unchecked). The user can check the box to select the option, or uncheck it to deselect the option. If necessary, the component can have three states, such as On, Off and Grayed, to do this, set the AllowGrayed component property to True.

Class: TdbCheckBox

Properties

Property	Type	Description
sqlValue	String	Returns the component value, for use in SQL queries. In case of empty valu *example:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+Forr
Alignment	TAlignment	Controls the position of the check box's caption. Available values: taRightJu
AllowGrayed	Boolean	Determines whether a check box can be in a dimmed state. If AllowGrayed AllowGrayed is set to False, the check box has only two possible states: sele
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary f False property, or if the component is located on a parent component with example: if Form1.CheckBox1.CanFocus then Form1.CheckBox1.SetFocus;
Caption	String	Specifies a text string that identifies the control to the user.
Checked	Boolean	Specifies whether the button control is checked. If the AllowGrayed proper

Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes in
dbTable	String	Determines which database table a component belongs to.
dbField	String	Determines which field of the database table this component belongs to.
dbIncremSearch	String	Allows you to specify the name of the button on the current form with the the value of the component. The property is necessary to implement instan
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
Hint	String	Hint contains the text string that appears when the user moves the mouse
Name	String	The name of the component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves ov
State	TCheckBoxState	Indicates whether the check box is selected, cleared, or dimmed. Available
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrde
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allow
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
WordWrap	Boolean	Specifies whether the button text wraps to fit the width of the control.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	Description
procedure **SetFocus**	Gives the input focus to the component.

Events

Event	Description
OnClick	Occurs when the user clicks the component.
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a component.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a component.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a

DateTimePicker

Description

DateTimePicker is designed specifically for entering dates or/and times.

Class: TdbDateTimePicker

Properties

Property	Type	Description
sqlDateTime	String	Returns the date and time value of the component for use in SQL queri return NULL. *example:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+F
sqlDate	String	Returns the date value of the component for use in SQL queries. The pr NULL. *пример:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+Fo
sqlTime	String	Returns the time value of the component for use in SQL queries. The pr NULL. *пример:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+Fo
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessa property, or if the component is located on a parent component with th example: if Form1.DateTimePicker.CanFocus then Form1.DateTimePicker1
Checked	Boolean	Indicates whether the check box next to the date or time is selected. Se
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passe
DateTime	TDateTime	Indicates the date that is marked on the calendar. More info about TDa
dbFilter	String	Makes sense when the component is used together with the button with
dbTable	String	Determines which database table a component belongs to.
dbField	String	Determines which field of the database table this component belongs t
dbIncremSearch	String	Allows you to specify the name of the button on the current form with th value of the component. The property is necessary to implement instant
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and tir
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
Format	String	Specify format for date-time string. More info.
TimeFormat	String	Makes sense when Kind property = DateTime. Allows you to set the for
Hint	String	Hint contains the text string that appears when the user moves the mou
Kind	TAdvDateTimeKind	Determines whether the component is a date selector, a time selector c
MaxDate	TDateTime	Indicates the maximum date to which users can scroll the calendar. Mor
MinDate	TDateTime	Indicates the minimum date that can be selected. More info about TDat
Name	String	The name of the component.
ShowCheckbox	Boolean	Displays a check box next to the date or time.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOr

TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to a
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component rela
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relati
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	Description
procedure **OpenCalendar**	Opens the drop-down calendar.
procedure **SetFocus**	Gives the input focus to the component.

Events

Event	Description
OnChange	Occurs when a date or time is entered.
OnClick	Occurs when the user clicks the component.
OnCloseUp	Occurs when the drop-down calendar closes.
OnDropDown	Occurs when the user opens the drop-down calendar by clicking the arrow at the right of the co
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.

Calendar

Description

Calendar is a component that displays the month calendar of the specified year.

Class: TdbComboBox

Properties

Property	Type	Description

sqlDate	String	Returns the date value of a component for use in SQL queries. The prop *example:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+F
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessa property, or if the component is located on a parent component with th example: if Form1.Calendar1.CanFocus then Form1.Calendar1.SetFocus;
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passe
Date	TDateTime	Indicates the date that is marked on the calendar. More info about TDa
dbFilter	String	Makes sense when the component is used together with the button with
dbTable	String	Determines which database table a component belongs to.
dbField	String	Determines which field of the database table this component belongs t
dbIncremSearch	String	Allows you to specify the name of the button on the current form with t value of the component. The property is necessary to implement instant
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and tir
EndDate	TDateTime	It makes sense when the MultiSelect property = True. Indicates the last
Focused	Boolean	Determines whether the control has input focus.
Hint	String	Hint contains the text string that appears when the user moves the mou
MaxDate	TDateTime	Indicates the maximum date to which users can scroll the calendar. Mor
MinDate	TDateTime	Indicates the minimum date that can be selected. More info about TDat
Name	String	The name of the component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves
ShowToday	Boolean	Specifies whether today's date is shown below the calendar.
ShowTodayCircle		Specifies whether today's date is circled on the calendar.
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOr
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to a
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Visible	Boolean	Specifies whether the component appears onscreen.
WeekNumbers	Boolean	Specifies whether week numbers are shown to the left of the calendar.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component rela
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relati
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	**Description**
procedure **SetFocus**	Gives the input focus to the component.

Events

Event	**Description**
OnClick	Occurs when the user clicks the component.
OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the co
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.

OnGetMonthBoldInfo	The event allows you to highlight certain days in the calendar. More info.
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.

OnGetMonthBoldInfo

Description

The event allows you to highlight certain days in the calendar.

```
procedure OnGetMonthBoldInfo (Sender: TObject; Month, Year: Cardinal; var
MonthBoldInfo: Cardinal);
```

Examples

```
// highlight days in the calendar, if January 2020 is shown
procedure Form1_MonthCalendar1_OnGetMonthBoldInfo (Sender: TObject; Month,
Year: Cardinal; var MonthBoldInfo: Cardinal);
begin
    if (Month=1) and (Year=2020) then
        TdbMonthCalendar(Sender).BoldDays([1,3,4,6,8,10], MonthBoldInfo);
end;



// Highlighting days in the calendar that are present in the database table
procedure Form1_MonthCalendar1_OnGetMonthBoldInfo (Sender: TObject; Month,
Year: Cardinal; var MonthBoldInfo: Cardinal);
var
    AStr: array of string;
    AByte: Array of byte;
    sMonth, s: string;
    i, c: integer;
begin
    // getting the days from database to be highlighted in the calendar
    sMonth := IntToStr(Month);
    if Length(sMonth)=1 then sMonth := '0' + sMonth;
    s := SQLExecute( 'SELECT group_concat(strftime(''%d'', "DateField"), ",")
FROM booking WHERE strftime(''%m.%Y'', "DateField"
="'+sMonth+'.'+IntToStr(Year)+'"' );

    if s <> '' then
    begin
        AStr := SplitString(s, ','); // Convert the string with days into an
array AStr
        SetLength(AByte, Length(AStr)); // set the length of the array AByte
        // convert an array of strings into a numeric array
        c := Length(AByte)-1;
        for i := 0 to c do
            if ValidInt(AStr[i]) then AByte[i] := StrToInt(AStr[i]) else
AByte[i] := 0;

        TdbMonthCalendar(Sender).BoldDays(AByte, MonthBoldInfo); // pass an
array of days to the component that you want to select
    end;
end;
```

ComboBox

Description

The component is used to show/select the record.

Class: TdbComboBox

Properties

Property	Type	Description
sqlValue	String	Returns the id of the selected record in the component, for use in SQL qu *example:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+Forr
dbItemID	Integer	The identifier of the selected record in the component. The identifier corre
dbSortField	String	The database field by which the records in the component will be sorted.
dbSortAsc	Boolean	If True, the records will be sorted in ascending order, otherwise in descen
dbFilter	String	Allows you to set an additional filter that will be used to fill the component
dbForeignKey	String	Specifies which external key of the database table this component belongs
dbField	String	Determines which field of the database table this component belongs to.
dbIncremSearch	String	Allows you to specify the name of the button on the current form with the value of the component. The property is necessary to implement instant se
dbSQL	String	Contains the SQL query that was used when the SQLExecute method was c
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary property, or if the component is located on a parent component with those example: if Form1.ComboBox1.CanFocus then Form1.ComboBox1.SetFocus;
Color	TColor	Specifies the background color of the control. More info.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes ir
DroppedDown	Boolean	Indicates whether the drop-down list is currently displayed.
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
GetCount	Integer	Allows you to get the number of records in a component.
HideTextIfNotExists	Boolean	Makes sense if the Searchable property = True. Set HideTextIfNotExists =
Hint	String	Hint contains the text string that appears when the user moves the mouse
ItemIndex	Integer	Specifies the index of the selected item. The first item in the list has index (

Items	TStringList	Provides access to the list of items (strings) in the list portion of the combo
Items[i]	String	Line-by-line access to the list of records. Example: ShowMessage(Form1.C
ItemsChecked[i]	Boolean	Makes sense if the MultiSelect = True property allows you to read or chec
MultiSelect	Boolean	Enables multiple selection of records in the component. Applies when usin
Name	String	The name of the component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves ov
SearchableType	TSearchType'	It makes sense if the Searchable property = True. Search Type. Values: stE
SelectedCount	Integer	Makes sense if the MultiSelect property = True. Returns the number of sel
SelLength	Integer	Specifies the length, in characters, of the selected text in the edit portion o
SelStart	Integer	Specifies the position of the first selected character in the edit portion of t
Sorted	Boolean	Determines whether the records in the component will be sorted. By defau change the field to be used for sorting using the dbSortField property. You
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrde
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allow
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Text	String	Contains a text string associated with the component.
TextHint	String	Specifies the text that is displayed as a text watermark in the edit box of th
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	Description
procedure **dbAddRecord** (id: integer; text: string)	Adds a record to the component and assigns it the specifi
procedure **dbEditRecord** (id: integer; text: string)	Edits the record in the component with the specified id. Th
procedure **dbDeleteRecord** (id: integer)	Deletes the record from the component with the specified
function **dbGetFieldValue** (FieldName: string): variant	The function allows you to get the value of the specified fie in the component, the function returns Unassigned.
function **dbIndexToID** (index: integer): integer	The function allows you to get the id of a record by specify
procedure **dbSQLExecute** (sql: string)	The procedure allows you to fill a component with data fro property.
procedure **dbUpdate**	Forces the data in the component to be updated. Usually t
procedure **Clear**	Clears the contents of the component.
procedure **DoOnChange**	Forcibly executes the OnChange event if it was defined for
procedure **SetAllCheckBoxes** (const Checked: boolean)	It makes sense if the MultiSelect property = True. Changes
procedure **SetFocus**	Gives the input focus to the component.

Events

Event	Description
OnChange	Occurs when the user changes the text displayed in the edit region.
OnClick	Occurs when the user clicks the component.
OnCloseUp	Occurs when the drop-down list closes up due to some user action.

OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the c
OnDropDown	Occurs when the drop-down list closes up due to some user action.
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnDropFiles	Occurs when user tries to drag and drop a file from explorer to a form. More info.

TableGrid

Description

The component is used to output records from the database in tabular form.

Class: TdbStringGridEx

Properties

Property	**Type**	**Description**
dbCustomOrderBy	String	Allows you to specify sorting. For example, you can specify sorting by two или tablename.fieldname1 ASC, tablename.fieldname2 DESC
dbFilter	String	Allows you to set an additional filter that will be used to populate the com
dbGeneralTable	String	Contains the name of the main database table; it is from this table that yo sqlValue property. The property can also contain an external key, for exam
dbGetSqlStatement	String	The property allows you to get the last SQL query that was used to popul
dbIncremSearch	String	Allows you to specify the name of the button on the current form with the the value of the component. The property is necessary to implement instar
dbItemID	Integer	The identifier of the selected record in the component. The identifier corre
dbLimit	Integer	Allows you to set the maximum number of records that will be populated
dbListFieldsNames	String	Contains header names for columns separated by commas.
dbOffSet	Integer	Allows you to set an offset when retrieving records from the database. M
dbParentTable	String	Contains the name of the parent database table when the component disp
dbParentTableId	Integer	Contains the identifier (id field) of the parent record in the parent table (d
dbPopupMenu	TPopupMenu	Provides access to the component's popup menu. More info.
dbSortAsc	Boolean	If True, the records will be sorted in ascending order, otherwise in descen dbCustomOrderBy property.
dbSortField	String	The database field by which the records in the component will be sorted. dbCustomOrderBy property.
dbSQL	String	Contains an SQL query if the component was populated with data using th will be executed when calling the dbSQLExecute method. More info.
sqlValue	String	Returns the id of the selected record in the component for use in SQL que *пример:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+For
AllowCreate	Boolean	It is responsible for the ability to create new records directly in the compo

AllowCreateEmpty	Boolean	It is responsible for the ability to create new empty records.
AllowEdit	Boolean	It is responsible for the ability to edit records directly in the component.
AllowDelete	Boolean	It is responsible for the ability to delete records directly in the component
SecondClickEdit	Boolean	Determines whether or not you need to double-click a cell to edit a recor
BorderStyle	TBorderStyle	Determines whether the component has a border. Available values: bsSing
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary property, or if the component is located on a parent component with thos example: if Form1.TableGrid1.CanFocus then Form1.TableGrid1.SetFocus;
Canvas	TCanvas	A class that allows you to draw on a component. More info.
Cell[x,y]	TCell	Property to access additional properties of the specified component cell (
Cells[x,y]	String	Property for accessing the text content of a cell (x - column, y - row).
ClientWidth	Integer	The width of the client part of the component (i.e. excluding borders).
ClientHeight	Integer	The height of the client part of the component (i.e., excluding the borders)
Color	TColor	The background color of the component. More info.
Columns	TNxColumns	Property to access additional column properties and methods. More info.
Columns[i]	TNxCustomColumn	Property to access the properties of a specified column. More info.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes i
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and time
FixedCols	Integer	Sets the number of fixed columns that will not move when scrolling horizo
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
HeaderSize	Integer	Gets or sets size of columns headers.
Hint	String	Hint contains the text string that appears when the user moves the mouse
HorzScrollBar	TNxScrollBar	Property to access additional scrollbar properties and methods. More inf
LastAddedRow	Integer	Returns the index of the last added line. Read-only.
Name	String	The name of the component.
Options	TGridOptions	Additional component settings. More info.
Parent	TWinControl	The parent component on which this component is placed.
RowCount	Integer	Gets total number of rows.
RowSize	Integer	Gets or sets default size (height) of rows in grid.
Row[i]	TRow	Access to additional properties of a row by its index. More info.
RowVisible[i]	Boolean	Determines the visibility of the row by its index.
Selected[i]	Boolean	Gets or sets specified row's selected state.
SelectedRow	Integer	Gets or sets selected Row's Index.
SelectedColumn	Integer	Gets or sets Index of selected Column.
SelectedCount	Integer	Gets number of selected rows.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves o
SlideSize	Integer	Gets or sets size (height) of single slide. Имеет смысл, когда свойство ко
SortedColumn	TNxCustomColumn	Reference to the sorted column.
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrde
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allo
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
VertScrollBar	TNxScrollBar	Property to access additional scrollbar properties and methods. More inf
Visible	Boolean	Specifies whether the component appears onscreen.
VisibleRows	Integer	Gets number of visible rows.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.

Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	**Description**
function **dbIndexToID** (index: Integer): integer	Allows you to get the record ID (database t
function **dbUpdate**: String	Forcibly updates data in the component. As used to access the database.
function **AddRow** (Count: Integer = 1): Integer	Adds the specified number of rows to the c
procedure **BeginUpdate**	Called before performing a large number o
procedure **BestFitColumns** (BestFitMode: TBestFitMode = bfCells)	The method automatically adjusts the width
procedure **BestFitRow** (const Index: Integer)	The method automatically adjusts the heigh
procedure **CalculateFooter** (VisibleOnly: Boolean = False)	Calculates footer of the component.
procedure **ClearRows**	Clears rows of of the component.
procedure **DeleteRow** (Index: Integer)	Deletes specified row. The record from the
function **ExportToExcel** (FileName: string = ''; ExcelVisible: boolean = True; FirsRowColumns: boolean = True): Variant	Exports data from the component to Excel.
function **ExportToLibreCalc** (FirsRowColumns: boolean = True): Boolean	Exports data from the component to Open
procedure **EndUpdate**	See BeginUpdate
function **GetRowAtPos** (X, Y: Integer): Integer	Allows you to get the row index by coordin
function **GetColumnAtPos** (X, Y: Integer): TNxCustomColumn	Allows you to get a link to a column by coo
procedure **InsertRow** (Pos: Integer; Count: Integer = 1)	Inserts single row at specified position. This
procedure **MoveRow** (FromPos, ToPos: Integer)	Moves row from specified position to anot
procedure **SaveToTextFile** (const FileName: String; Separator: Char = ','; MultiLineSeparator: Char = '|')	Saves the contents of the component to a t
procedure **SaveToHtml** (FileName: String; SaveHeaders: boolean = True; AllRows: boolean = False; CreateStyleSheet: boolean = True; SaveFooter: boolean = False; SaveCaption: boolean = False)	Saves the content of the component in an H
procedure **LoadFromTextFile** (const FileName: String; Separator: Char = ','; MultiLineSeparator: Char = '|'; StartRow: Integer = 0)	Loads text file data into the component. Th
procedure **ScrollToRow** (index: integer)	Moves the scroll in the component so that
procedure **SelectAll**	Selects all rows in the component. For the r
procedure **SelectRange** (FromRow, ToRow: Integer; Value: Boolean)	Selects (or deselects, if Value = False) the s
procedure **SetFocus**	Gives the input focus to the component.
procedure **SwapRows** (FromPos, ToPos: Integer)	Swaps positions of two rows.

Events

Event	**Description**
OnApplyEditText (Sender: TObject; ACol, ARow: Integer; var Value: String)	Occurs when you finish editin
OnAfterEdit (Sender: TObject; ACol, ARow: Integer; Value: String)	Occurs when the cell editing
OnAfterRowMove (Sender: TObject; FromPos, ToPos: Integer)	Occurs when the user has su
OnAfterSort (Sender: TObject; ACol: Integer)	Occurs after the column is so
OnBeforeEdit (Sender: TObject; ACol, ARow: Integer; var Accept: Boolean)	Occurs before the cell enters

OnCellClick (Sender: TObject; ACol, ARow: Integer)	Occurs when the user clicks o
OnCellDoubleClick (Sender: TObject; ACol, ARow: Integer)	Occurs when the user double
OnChange (Sender: TObject)	Occurs after the component
OnClick (Sender: TObject)	Occurs when the user clicks t
OnColumnResize (Sender: TObject; ACol: Integer)	Occurs when the user resize
OnDoubleClick (Sender: TObject)	Occurs when the user double
OnEditAccept (Sender: TObject; ACol, ARow: Integer; Value: String; var Accept: Boolean)	Occurs when the user has fin
OnEnter (Sender: TObject)	Occurs when a component re
OnExit (Sender: TObject)	Occurs when the input focus
OnFooterClick (Sender: TObject; ACol: Integer)	Occurs when the user clicks o
OnHeaderClick (Sender: TObject; ACol: Integer)	Occurs when the user clicks o
OnHeaderDoubleClick (Sender: TObject; ACol: Integer)	Occurs when the user double
OnInputAccept (Sender: TObject; var Accept: Boolean)	Occurs before a new record
OnInputSelectCell (Sender: TObject; ACol: Integer)	Occurs when the user has mo
OnKeyDown (Sender: TObject; var Key: Word; Shift, Alt, Ctrl: boolean)	Occurs when a user presses
OnKeyPress (Sender: TObject; var Key: Char)	Occurs when a key is pressed
OnKeyUp (Sender: TObject; var Key: Word; Shift, Alt, Ctrl: boolean)	Occurs when the user release
OnLoadProgress (Sender: TObject; ACol, ARow: Integer)	Occurs when a text file is loa
OnMouseDown (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)	Occurs when the user presse
OnMouseEnter (Sender: TObject)	Occurs when the user moves
OnMouseLeave (Sender: TObject)	Occurs when the user moves
OnMouseMove (Sender: TObject; Shift, Alt, Ctrl: boolean; X, Y: Integer)	Occurs when the user moves
OnMouseUp (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)	Occurs when the user release
OnResize (Sender: TObject)	Occurs when a component is
OnRowMove (Sender: TObject; FromPos, ToPos: Integer; var Accept: Boolean)	Occurs when the user has mo
OnSortColumn (Sender: TObject; ACol: Integer; Ascending: Boolean)	Occurs before the column is
OnDropFiles (Sender: TObject; ArrayOfFiles: array of string; X, Y: Integer)	Occurs when user tries to dra

property dbLimit: Integer

Description

Allows you to set the maximum number of records that will be populated in the component.

This property works only when the component displays data with the "Search" action button or when the component option "Show all records from table" is used.

To remove the limit on the number of records, set the value to 0.

Example

```
Form1.TableGrid1.dbLimit := 1000;
Form1.TableGrid1.dbLimit := 0; // removes the limit on the number of records
```

property dbOffSet: Integer

Description

Allows you to set an offset when retrieving records from the database.

Typically, this property is used to implement page-by-page output of records in a component.

To remove the offset for records, set it to 0.

Example

```
Form1.TableGrid1.dbOffSet := 1000;
Form1.TableGrid1.dbOffSet := 0; // removes offset of records
```

A project with an example of page-by-page output of records.

property dbSQL: string

Description

Contains an SQL query if the component was populated with data using the button with the "SQL query" action.

It also allows you to set your own SQL query that will be executed when the dbSQLExecute method is called.

Example

```
procedure Form1_Button1_OnClick (Sender: string; var Cancel: boolean);
begin
    // include the id field into SQL query, if the possibility to edit or
delete records is necessary
    // include "$autoinc" into SQL query if a column with sequential numbering
is required
    Form1.TableGrid1.dbSQL:='SELECT id, "$autoinc", lastname, firstname,
salary FROM employees';

    //Form1.GridEmployees.dbParentTable := 'ParentTable'; // optionally
    //Form1.GridEmployees.dbParentTableId := 1; // optionally

    // optional, in case of complex SQL query, specify the main database table
manually,
    // to which the id field specified in the SQL query in the dbSQL property
will belong
    Form1.TableGrid1.dbGeneralTable := 'employees';
    // specify header names for columns, separated by commas,
    // if it is necessary to hide a column in a component, specify the name
delete_col, as a rule, it is useful to hide the id field
    Form1.TableGrid1.dbListFieldsNames :='delete_col,#,name2,name3,name4';
    Form1.TableGrid1.dbSQLExecute; // Execute an SQL query
end;
```

property dbPopupMenu: TPopupMenu

Description

Provides access to the component's popup menu.

Allows you to configure the popup menu.

Examples

```
// hide menu item
Form1.TableGrid1.dbPopupMenu.Items[0].Visible := False;


// disable menu item
Form1.TableGrid1.dbPopupMenu.Items[0].Enabled := False;


// programmatically click on the first menu item (numbering starts from 0)
Form1.TableGrid1.dbPopupMenu.Items[0].Click;



// renaming menu items
procedure Form1_OnShow (Sender: TObject; Action: string);
begin
    Form1.TableGrid1.dbPopupMenu.Items[0].Caption := 'Show record 2';
    Form1.TableGrid1.dbPopupMenu.Items[1].Caption := 'Delete record 2';
    Form1.TableGrid1.dbPopupMenu.Items[3].Caption := 'Copy cell 2';
    Form1.TableGrid1.dbPopupMenu.Items[4].Caption := 'Copy 2';
    Form1.TableGrid1.dbPopupMenu.Items[5].Caption := 'Copy all 2';
    Form1.TableGrid1.dbPopupMenu.Items[7].Caption := 'Find 2';
end;



// adding menu items and submenu
procedure Form1_OnShow (Sender: TObject; Action: string);
var
   SubMenu: TMenuItem;
   MenuItem: TMenuItem;
begin
   SubMenu := TMenuItem.Create (Form1);
   SubMenu.Caption := 'SubMenu';
   MenuItem := TMenuItem.Create (Form1);
   MenuItem.Caption := 'Item';
   MenuItem.OnClick := @MenuClick1;
   Form1.TableGrid1.dbPopupMenu.Items.Insert(0, SubMenu);
   Form1.TableGrid1.dbPopupMenu.Items[0].Add(MenuItem);
end;

procedure MenuClick1;
begin
    ShowMessage('Hello from PopupMenu');
end;
```

property Cell[x,y]: TCell

Description

Property to access additional properties of the specified component cell (x - column, y - row).

The class has the following properties

Property	Description
AsBoolean: Boolean	Gets or sets value of item in Boolean type.
AsDateTime: TDateTime	Gets or sets value of item in TDateTime type.
AsFloat: Double	Gets or sets value of item in Double type.
AsInteger: Integer	Gets or sets value of item in Integer type.
Color: TColor	Gets or sets Cell Color.
Empty: Boolean	Determines whether the cell contains any value.
Hint: string	Gets or sets Cell Hint.
ObjectReference: TObject	Gets or sets Reference to TObject object.
FontStyle: TFontStyles	Gets or sets Cell Font Style. Available values: fsBold+fsItalic+fsUnderline+fsStrikeOut
Tag: Integer	Allows you to assign a number for your own needs.
TextColor: TColor	Specifies the font color for the cell.

Example

```
// changes the color of the first cell
procedure Form1_TableGrid1_OnChange (Sender: TObject);
begin
    Form1.TableGrid1.Cell[0,0].TextColor := clRed;
end;
```

property Columns: TNxColumns

Description

Property for accessing additional column properties and methods.

The class has the following properties

Properties and methods	Description
procedure **Add** (Source: TNxCustomColumn): TNxCustomColumn	Adds a new column with the specified class.
procedure **Clear**	Destroys all columns.
property **Count**: Integer	Returns the number of columns.
procedure **Delete** (index: integer)	Deletes the column with the specified index, the c
function **InsertCheckBoxColumn**(Pos: Integer): TNxCustomColumn	Inserting a column with a CheckBox (TNxCheckBox
function **InsertGraphicColumn**(Pos: Integer): TNxCustomColumn	Inserting a graphical column (TNxGraphicColumn)
function **InsertTreeColumn**(Pos: Integer): TNxCustomColumn	Inserting a tree column (TNxTreeColumn) in the sp
property **LastAdded**: TNxCustomColumn	Returns a reference to the last added column.

Examples

```
// As a rule, the creation of additional columns should happen in the event of
the OnChange component

// adding a column of the specified class, available classes:
// TNxTextColumn, TNxNumberColumn, TNxDateColumn, TNxTimeColumn,
TNxCheckBoxColumn, TNxListColumn, TNxGraphicColumn, TNxTreeColumn
try
   Form1.TableGrid1.Columns.Add(TNxTextColumn);
except
end;
Form1.TableGrid1.Columns.LastAdded.Color := clWhite;



// Inserting a column with a CheckBox (TNxCheckBoxColumn) at the specified
position, column numbering starts from 0.
Form1.TableGrid1.Columns.InsertCheckBoxColumn(0);
```

property Columns[i]: TNxCustomColumn

Description

Property to access the properties of the specified column.

The class has the following properties

Property	Type	Description
Alignment	TAlignment	Sets the alignment of the column content. Available values: taCenter, ta
Enabled	Boolean	Determines the availability of the column to the user.
Footer	TColumnFooter	The property is responsible for the footer of the component. More inf
Header	TColumnHeader	The property is responsible for the headers of the component. More i
Options	TColumnOptions	Additional settings for the column. More info.
SlideAnchors	TAnchors	Makes sense when the GridStyles component property = gsSlides. Exa
SlideBounds	TNxSlideBounds	Makes sense when the GridStyles component property = gsSlides. Exa

SlideCaption	String	Makes sense when the GridStyles component property = gsSlides. Exa
SlideCaptionLocation	TSlideCatpionLocation	Makes sense when the GridStyles component property = gsSlides. Exa
Sorted	Boolean	Sorts the column.
SortKind	TSortKind	Specifies whether the column is sorted in ascending or descending orc
SortType	TSortType	Defines the sort type for the column. Values: stAlphabetic, stBoolean, s
Tag	Integer	Allows you to assign a number to the column for your own needs.
TagString	String	Allows you to assign a string to a column for your own needs.
VerticalAlignment	TVerticalAlignment	Sets the vertical alignment of the column content. Available values: vaT
Visible	Boolean	Determines the visibility of the column in the component.
Width	Integer	Sets the width of the column in the component.
WrapKind	TWrapKind	Sets the behavior of text that does not fit in the cell for a given column

property Footer: TColumnFooter

Description

This property is responsible for the footer of the component.

The class has the following properties

Property	Type	Description
Alignment	TAlignment	Sets the alignment of the footer content in the column. Available values: ta(
Caption	String	Footer text.
Color	TColor	Background Color.
FormulaKind	TFormulaKind	Formula for the calculation. Available values: fkNone, fkAverage, fkCount, fk
FormatMask	String	Allows you to set the format for numbers, text or date/time.
FormatMaskKind	TFormatMaskKind	Defines the way the mask in the FormatMask property is handled. Available
TextAfter	String	Specifies the text before the calculated value.
TextBefore	String	Specifies the text after the calculated value.

Example

```
// format the value in the basement for the columns of the numeric type (REAL,
CURRENCY, INTEGER)
// more details about using formatting
http://docwiki.embarcadero.com/Libraries/XE3/en/System.SysUtils.FormatFloat
procedure Form1_TableGrid1_OnChange (Sender: TObject);
begin
    TNxNumberColumn(Form1.TableGrid1.Columns[0]).Footer.TextBefore := Price:
';
    TNxNumberColumn(Form1.TableGrid1.Columns[0]).Footer.TextBefore := '$';
    TNxNumberColumn(Form1.TableGrid1.Columns[0]).Footer.FormatMaskKind :=
mkFloat;
    TNxNumberColumn(Form1.TableGrid1.Columns[0]).Footer.FormatMask :=
'#,##0.00';
end;
```

property Header: TColumnHeader

Description

This property is responsible for the footer of the component.

The class has the following properties

Property	Type	Description
Alignment	TAlignment	Sets the alignment of the header in the column. Available values: taCenter,
Caption	String	Title text.
Color	TColor	Header background color. Ignored if component property EnableVisualSt
DisplayMode	TDisplayMode	Defines the mode of displaying text and graphics in the header. Available
Glyph	TBitmap	Allows you to put a bmp image in the header.
Hint	String	A tooltip for the header.
MultiLine	Boolean	Allows you to use multiple strings in the header.
Orientation	THeaderOrientation	Defines the orientation of the header. Available values: hoHorizontal, hoV

Example

```
// place the picture in the header
procedure Form1_TableGrid1_OnChange (Sender: TObject);
begin
    Form1.TableGrid1.Columns[0].Header.DisplayMode := dmTextAndImage;
    // get the image from Image1 placed on the form, the image must be in bmp
format
    Form1.TableGrid1.Columns[
0].Header.Glyph.Assign(Form1.Image1.Picture.Bitmap);
end;
```

property Options: TColumnOptions

Description

Additional settings for the column.

The set has the following values.

Value	Description
coAutoSize	Not used. For automatic column widths, use the BestFitColumns method.
coCanClick	Determines whether the header of a given column can be clicked to sort.

coCanInput	Makes sense if the TableGrid.Options property has the value goInput. Allows you to enter a value
coCanSort	Determines whether sorting is executed when you click on the header of a given column.
coDisableMoving	Disables drag-and-drop of columns.
coEditing	Specifies the ability to edit the text in the rows of a given column.
coEditorAutoSelect	Determines whether text is automatically selected when the input cursor is set in a cell of a given
coFixedSize	Disables the resizing of this column.
coShowTextFitHint	Determines whether a hint will be shown if the contents of the cell do not fit.

Example

```
// As a rule, it is necessary to change column settings in the OnChange event
procedure Form1_TableGrid1_OnChange (Sender: TObject);
begin
    Form1.TableGrid1.Columns[0].Options := Form1.TableGrid1.Columns[0].Options
- coCanSort; // disables sorting for the first column
    Form1.TableGrid1.Columns[1].Options := Form1.TableGrid1.Columns[1].Options
+ coFixedSize; // Disables the ability to resize the second column
end;
```

property HorzScrollBar: TNxScrollBar

Description

A property for accessing additional scroll properties and methods.

The class has the following properties and methods

Properties and methods	Description
property **AutoHide**: Boolean	Determines whether to hide the scrollbar if it is not needed.
procedure **BeginUpdate**	The method disables the scroll update. After the necessary changes, you need t
procedure **EndUpdate**	The method enables a scroll update.
property **Enabled**: Boolean	Determines the availability of the scroll to the user.
procedure **First**	The method sets the scroll position to the beginning.
function **IsFirst**: Boolean	The function returns True if the scroll is at the beginning position.
function **IsLast**: Boolean	The function returns True if the scroll is at the end position.
procedure **Last**	The method sets the scroll position to the end.
property **ManualScroll**: Boolean	Allows you to enable manual scrolling. In this case, the scrolling will need to be i
property **MoveBy**(Distance: Integer)	The method moves the scroll position by the specified value. To move the scroll
procedure **Next**	The method moves the scroll position to the right (move step can be set in pixe
property **PageSize**: Integer	Returns the number of visible rows.
procedure **PageDown**	The method moves the scroll position, just like when you press the PageDown ke
procedure **PageUp**	The method moves the scroll position, just like when you press the PageUp keyb

property **Position**: Integer	The property allows you to get or set a scroll position.
procedure **Prior**	The method moves the scroll position to the left (move step can be set in pixels
property **SmallChange**: Integer	Allows you to set the step for moving the scroll in pixels.
property **Visible**: Boolean	Determines the visibility of the scroll bar.

property VertScrollBar: TNxScrollBar

Description

A property for accessing additional scroll properties and methods.

The class has the following properties and methods

Properties and methods	**Description**
property **AutoHide**: Boolean	Determines whether to hide the scrollbar if it is not needed.
procedure **BeginUpdate**	The method disables the scroll update. After the necessary changes, you need t
procedure **EndUpdate**	The method enables a scroll update.
property **Enabled**: Boolean	Determines the availability of the scroll to the user.
procedure **First**	The method sets the scroll position to the beginning.
function **IsFirst**: Boolean	The function returns True if the scroll is at the beginning position.
function **IsLast**: Boolean	The function returns True if the scroll is at the end position.
procedure **Last**	The method sets the scroll position to the end.
property **ManualScroll**: Boolean	Allows you to enable manual scrolling. In this case, the scrolling will need to be
property **MoveBy**(Distance: Integer)	The method moves the scroll position by the specified value. To move the scroll
procedure **Next**	The method moves the scroll position to the down (move step can be set in pix
property **PageSize**: Integer	Returns the number of visible rows.
procedure **PageDown**	The method moves the scroll position, just like when you press the PageDown k
procedure **PageUp**	The method moves the scroll position, just like when you press the PageUp keyb
property **Position**: Integer	The property allows you to get or set a scroll position.
procedure **Prior**	The method moves the scroll position to the up (move step can be set in pixels
property **SmallChange**: Integer	Allows you to set the step for moving the scroll in pixels.
property **Visible**: Boolean	Determines the visibility of the scroll bar.

property Options: TGridOptions

Description

Additional component settings.

You can combine the settings using the listed values.

Value	**Description**
goArrowKeyExitEditing	Exit cell editing by pressing up, down, left, right buttons
goCanHideColumn	The user will be able to hide columns with the mouse.
goDisableColumnMoving	Disables the user to move columns with the mouse.

goDisableKeys	Disables moving between rows with arrows on the keyboard.
goEscClearEdit	When editing a cell, allows you to clear it by pressing Esc.
goFooter	Makes the component's footer visible.
goGrid	Shows horizontal and vertical lines.
goHeader	Shows the headings for the columns.
goIndicator	Shows the indicator of the selected row.
goInput	Shows the input bar.
goLockFixedCols	Disables moving fixed columns with mouse (FixedCols).
goMultiSelect	Allows you to select multiple records using the Ctrl or Shift key.
goRowResizing	Allows you to change the height of rows with the mouse (goIndicator must also be activate
goRowMoving	Allows you to move rows with the mouse.
goSecondClickEdit	Editing a cell with a double click of the mouse.
goSelectFullRow	Selects a whole row, otherwise individual cells will be selected.

The default settings are as follows:
goDisableColumnMoving, goGrid, goHeader, goSecondClickEdit, goSelectFullRow

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
begin
    // disable lines in the component
    Form1.GridSearch.Options := Form1.GridSearch.Options - goGrid;

    // Set the necessary settings for the component
    Form1.GridSearch.Options := goDisableColumnMoving + goGrid + goHeader +
goSecondClickEdit + goSelectFullRow + goMultiSelect;
end;
```

property Row[i]: TRow

Description

Property to access additional properties of the specified row.

The class has the following properties

Property	**Description**
ChildCount: Integer	Number of child elements (makes sense if there is a TNxTreeColumn).
Expanded: Boolean	Whether the row is expanded to show the children (makes sense if there is a TNxTreeColu
HasChildren: Boolean	Does the row have children (makes sense if there is a TNxTreeColumn).
ID: Integer	Contains record ID (id field from database).
ImageIndex: Integer	Icon index from TImageList in property TNxTreeColumn(Form1.TreeView1.Columns[0]).Ima
Level: Integer	Contains the nesting level (makes sense if there is a TNxTreeColumn).
ParentRow: TRow	Reference to the parent row (makes sense if there is a TNxTreeColumn).
RowHeight: Integer	Gets or sets the height of a Row.
Selected: Boolean	Allows you to know if a row is selected or not.
Shown: Boolean	Allows you to know if the given row is visible (makes sense if there is a TNxTreeColumn).

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
begin
   // show the ID of the specified row
   ShowMessage( Form1.TableGrid1.Row[0].ID );
end;
```

procedure BestFitColumns(BestFitMode: TBestFitMode = bfCells)

Description

The method allows you to automatically adjust the width of the columns.

As a rule, this method should be called in the event of the OnChange component.

You can specify the following values as a parameter:

bfCells - autosize width by cells contents.
bfBoth - autosize width by contents of cells and column titles.
bfHeader - autosize width by column headers.

Example

```
procedure Form1_TableGrid1_OnChange (Sender: TObject);
begin
    Form1.TableGrid1.BestFitColumns(bfBoth);
end;
```

procedure BestFitRow(const Index: Integer)

Description

The method automatically adjusts the height of the row, depending on the contents of the cells in this row.

As a rule, this method must be called in the event of the OnChange component.
Also, you must assign the WrapKind := wkWordWrap property to columns to allow wrap text in cells.

Example

```
// automatically adjust the height of all rows in the component
procedure Form1_TableGrid1_OnChange (Sender: string);
var
    i, c: integer;
```

```
begin
    c := Form1.TableGrid1.Columns.Count - 1;
    for i := 0 to c do
    begin
        Form1.TableGrid1.Columns[i].VerticalAlignment := taAlignTop; // set
the vertical alignment in the cells of the column
        Form1.TableGrid1.Columns[i].WrapKind := wkWordWrap; // enable the
possibility of wrapping the strings in the cells of the column
    end;

    c := Form1.TableGrid1.RowCount - 1;
    for i := 0 to c do Form1.TableGrid1.BestFitRow(i); // for each row we call
the method for auto height adjustment
end;


// also updates the height of the rows when you resize the columns
procedure Form1_TableGrid1_OnColumnResize (Sender: TObject; ACol: Integer);
var
    i, c: integer;
begin
    c := Form1.TableGrid1.RowCount - 1;
    for i := 0 to c do Form1.TableGrid1.BestFitRow(i); // for each row we call
the method for auto height adjustment
end;
```

procedure OnApplyEditText (Sender: TObject; ACol, ARow: Integer; var Value: String)

Description

Occurs when you finish editing a cell. Allows you to change the entered value.

When editing a cell, the following chain of events is triggered: OnBeforeEdit > **OnAplyEditText** > OnEditAccept > OnAfterEdit

In the parameters of this event there is a Value parameter that allows you to change the value entered by the user.

Example

```
procedure Form1_TableGrid1_OnApplyEditText (Sender: TObject; ACol, ARow:
Integer; var Value: String);
begin
   if Value = 'Hello' then Value := 'Bye'; // If the entered value is Hello,
then change it to Bye
end;
```

procedure OnAfterEdit(Sender: TObject; ACol, ARow: Integer; Value: String)

Description

Occurs when editing a cell is successfully completed.

When editing a cell, the following chain of events is triggered: OnBeforeEdit > OnAplyEditText > OnEditAccept > **OnAfterEdit**

Example

```
procedure Form1_TableGrid1_OnAfterEdit (Sender: TObject; ACol, ARow: Integer;
Value: String);
begin
    ShowMessage(Value); // display the value entered by the user in the cell
end;
```

procedure OnBeforeEdit(Sender: TObject; ACol, ARow: Integer; var Accept: Boolean)

Description

It is triggered before the cell enters edit mode. Allows you to disable editing.

When editing a cell, the following chain of events is triggered: **OnBeforeEdit** > OnAplyEditText > OnEditAccept > OnAfterEdit

The parameters of this event contain the Accept parameter that allows you to prevent editing of a cell.

Example

```
procedure Form1_TableGrid1_OnBeforeEdit (Sender: TObject; ACol, ARow: Integer;
var Accept: Boolean);
begin
   // Prevent editing columns 2 and 3 (numbering columns from 0)
   if (ACol = 3) and (ARow = 4) then Accept := False;

   // Cancel edit if you entered an empty value
   if Form1.TableGrid1.Cells[ACol, ARow] = '' then Accept := False;
end;
```

procedure OnCellClick(Sender: TObject; ACol, ARow: Integer)

Description

It is triggered when the user clicked on a cell.

The event contains parameters ACol and ARow, which contain the column number and row number, respectively, on which the user clicked.

Example

```
// show the contents of the cell clicked on by the user
procedure Form1_TableGrid1_OnCellClick (Sender: TObject; ACol, ARow: Integer);
begin
    ShowMessage( Form1.TableGrid1.Cells[ACol, ARow] );
end;
```

procedure OnEditAccept(Sender: TObject; ACol, ARow: Integer; Value: String; var Accept: Boolean)

Description

Occurs when the user has finished editing a cell, such as selecting another cell or pressing Enter. Allows you to reject the entered value.

When editing a cell, the following chain of events is triggered: OnBeforeEdit > OnAplyEditText > **OnEditAccept** > OnAfterEdit

The parameters of this event contain an Accept parameter that allows you to reject a value entered in a cell.

Example

```
procedure Form1_TableGrid1_OnEditAccept (Sender: TObject; ACol, ARow: Integer;
Value: String; var Accept: Boolean);
begin
   // prohibits entering empty values
   if Value = '' then Accept := False;

   // prohibits entering values shorter than 3 characters
   if Length(Value) < 3 then Accept := False;
end;
```

procedure OnInputAccept(Sender: TObject; var Accept: Boolean)

Description

Occurs before a new record is added from the input line. Allows you to cancel the creation of a record.

The parameters of this event include the Accept parameter, which allows you to cancel the creation of a new record.

Example

```
procedure Form1_TableGrid1_OnInputAccept (Sender: TObject; var Accept:
```

```
Boolean);
begin
   // prevents an add record if "123" is entered in the first column
   if Form1.TableGrid1.Columns[0].InputValue = '123' then Accept := False;
end;
```

procedure OnRowMove(Sender: TObject; FromPos, ToPos: Integer; var Accept: Boolean)

Description

Occurs when the user has moved a row to a new position. Allows you to cancel the move.

The parameters of this event include the Accept parameter, which allows you to cancel the move of the row.

Example

```
// prohibit moving the first line to the very end
procedure Form1_TableGrid1_OnRowMove (Sender: TObject; FromPos, ToPos:
Integer; var Accept: Boolean);
begin
    if (FromPos=0) and (ToPos = Form1.TableGrid1.RowCount-1) then
    begin
        Accept := False;
        ShowMessage('You cannot move the first line to the end.');
    end;
end;
```

Counter

Description

Allows you to assign a unique number to records.

Class: TdbEditCount

Properties

Property	Type	Description

sqlValue	String	Returns the value of a component, for use in SQL queries. The property va Currency = True, the escape quotes will be omitted. In case of an empty va *example:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES ('+For
Alignment	TAlignment	Determines how the text is aligned within the text edit control. Available va
AutoSelect	Boolean	Determines whether all the text in the edit control is automatically selected
BorderStyle	TBorderStyle	Determines whether the edit control has a single line border around the cl
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary b property, or if the component is located on a parent component with those example: if Form1.EditCounter1.CanFocus then Form1.EditCounter1.SetFocu
CharCase	TEditCharCase	Determines the case of the text within the edit control. Available values: ec
Color	TColor	Specifies the background color of the control. More info.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes i
dbTable	String	Determines which database table a component belongs to.
dbField	String	Determines which field of the database table this component belongs to.
dbIncremSearch	String	Allows you to specify the name of the button on the current form with the text for instant search.
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
GetTextLen	Integer	Returns the length of the component's text.
Hint	String	Hint contains the text string that appears when the user moves the mouse
MaxLength	Integer	Specifies the maximum number of characters the user can enter into the e
Name	String	The name of the component.
PasswordChar	String	Indicates the character, if any, to display in place of the actual characters t used: *
ReadOnly	Boolean	Determines whether the user can change the text of the edit component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves o
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrde
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allo
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Text	String	Contains a text string associated with the component.
TextHint	String	A hint or message to be displayed when the Text property is empty.
Value	Double	The numerical value of the component.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	Description
procedure **Clear**	Deletes all text from the edit component.
procedure **CopyToClipboard**	Copies the selected text in the edit component to the Clipboard.
procedure **CutToClipboard**	Copies the selected text to the Clipboard and then deletes the selection.
procedure **PasteFromClipboard**	Pastes the contents of the Clipboard into edit component, replacing the current sel
procedure **SelectAll**	Selects all text in the edit component.

procedure **SetFocus**	Gives the input focus to the component.

Events

Event	Description
OnChange	Occurs when the text for the edit component may have changed.
OnClick	Occurs when the user clicks the component.
OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the co
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.
OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a component.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a component.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a
OnDropFiles	Occurs when user tries to drag and drop a file from explorer to a form. More info.

DBFile

Description

The component is used to save the file to the database and retrieve it from the database.

Class: TdbFileToDatabase

Properties

Property	Type	Description
dbCopyTo	String	Allows you to specify where you want the file to be copied automatically. M
dbFileName	String	If the Type = LinkFile property, then the property returns the full path of th path relative to the database file location (only for SQLite). If Type = StoreF
dbFileIsChanged	Boolean	If the property returns True, then the file in the component has been chang
dbInitialDir	String	Allows you to set the default path for the open and save file dialog.
Alignment	TAlignment	Sets the text alignment. Available values: taCenter, taRightJustify, taLeftJusti
AutoSelect	Boolean	Determines whether all the text in the edit control is automatically selected
BorderStyle	TBorderStyle	Determines whether the edit control has a single line border around the cl
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary False property, or if the component is located on a parent component with example: if Form1.DBFile1.CanFocus then Form1.DBFile1.SetFocus;
CharCase	TEditCharCase	Determines the case of the text within the edit control. Available values: ecl

Color	TColor	Specifies the background color of the control. More info.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes ir
dbTable	String	Determines which database table a component belongs to.
dbField	String	Determines which field of the database table this component belongs to.
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
GetTextLen	Integer	Returns the length of the component's text.
Hint	String	Hint contains the text string that appears when the user moves the mouse
LeftButton	TEditButton	To access the left button properties (Enabled, Hint, Visible).
MaxLength	Integer	Specifies the maximum number of characters the user can enter into the ec
Name	String	The name of the component.
PopupMenu	TPopupMenu	Provides access to the component's popup menu from the right button.
ReadOnly	Boolean	Determines whether the user can change the text of the edit component.
RightButton	TEditButton	To access the properties of the right button (Enabled, Hint, Visible).
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves ov
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrde
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allo
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Text	String	Contains a text string associated with the component.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	**Description**
procedure **Clear**	Deletes all text from the edit component.
procedure **ClearEx**	Clears the data and text content of the component.
procedure **CopyToClipboard**	Copies the selected text in the edit component to the Clipboard.
procedure **CutToClipboard**	Copies the selected text to the Clipboard and then deletes the selection.
procedure **PasteFromClipboard**	Pastes the contents of the Clipboard into edit component, replacing the current sel
procedure **SelectAll**	Selects all text in the edit component.
procedure **SetFocus**	Gives the input focus to the component.

Events

Event	**Description**
OnChange	Occurs when you change the text in a component.
OnClick	Occurs when the user clicks the component.
OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the co
OnEnter	Occurs when a component receives the input focus.
OnExit	Occurs when the input focus shifts away from one component to another.

OnKeyDown	Occurs when a user presses any key while the form has focus.
OnKeyPress	Occurs when a key is pressed. Note that this procedure handles printable characters only.
OnKeyUp	Occurs when the user releases a key that was pressed.
OnLeftButtonClick	Occurs when the left button is pressed.
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a component.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a component.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a
OnRightButtonClick	Occurs when the right button is pressed.
OnDropFiles	Occurs when user tries to drag and drop a file from explorer to a form. More info.

DBImage

Description

The component is used to save images to the database. Supported formats: jpg, bmp, gif, png8, png24.

Class: TdbImageDataBase

Properties

Property	Type	Description
dbCopyTo	String	Allows you to specify where you want the file to be copied automatically. M
dbFileName	String	If the Type = LinkFile property, then the property returns the full path of th path relative to the database file location (only for SQLite). If Type = StoreF
dbImageIsChanged	Boolean	If the property returns True, it means that the picture file in the componen
dbInitialDir	String	Allows you to set the default path for the open and save file dialog.
dbShowButtons	Boolean	Allows you to hide the buttons on a component that appear when you hov
AutoSize	Boolean	Specifies whether the control sizes itself automatically to accommodate the
ButtonOpen	TToolButton	Access the properties of the "Open" button.
ButtonSave	TToolButton	Access the properties of the "Save" button.
ButtonDelete	TToolButton	Access the properties of the "Delete" button.
Center	Boolean	Indicates whether the image is centered in the image control.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes in
dbTable	String	Determines which database table a component belongs to.
dbField	String	Determines which field of the database table this component belongs to.
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer
Hint	String	Hint contains the text string that appears when the user moves the mouse
isEmpty	Boolean	It returns True if the component has a picture loaded, otherwise it returns
Name	String	The name of the component.
OpenDialogFilter	String	This property allows you to set a file filter for the dialog box for selecting *.jpg;*.jpeg|BMP files (*.bmp)|*.bmp|All files|*.*';
Picture	TPicture	Access to picture properties and methods.
Proportional	Boolean	Indicates whether the image should be changed, without distortion, so tha

		property
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves o
ShowButtonOpen	Boolean	Determines whether the "Open" button is displayed or not.
ShowButtonSave	Boolean	Determines whether the "Save" button is displayed or not.
ShowButtonDelete	Boolean	Determines whether the "Delete" button is displayed or not.
Stretch	Boolean	Indicates whether the image should be changed so that it exactly fits the b
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relativ
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	**Description**
procedure **Clear**	Clears the contents of the component.
function **CopyToClipboard**: Boolean	Copies the image to the clipboard.
procedure **LoadFromDatabase** (TableName, FieldName: string; id: integer)	Loads a picture from the database.

Events

Event	**Description**
OnClick	Occurs when the user clicks the component.
OnDoubleClick	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the c
OnMouseDown	Occurs when the user presses a mouse button with the mouse pointer over a component.
OnMouseEnter	Occurs when the user moves the mouse into a component.
OnMouseLeave	Occurs when the user moves the mouse outside of a component.
OnMouseMove	Occurs when the user moves the mouse pointer while the mouse pointer is over a component.
OnMouseUp	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a

TreeView

Description

It serves for output and creation of data in a hierarchical form (tree structure). An example of hierarchical data is the structure of a company.

Class: TdbTreeView

Properties

Property	Type	Description
dbFilter	String	Allows you to set an additional filter that will be used to populate the
dbForeignKey	String	Specifies which foreign key of the database table this component be
dbFieldParentID	String	Specifies the field that will be used to form the tree structure. The fie
dbGetSqlStatement	String	The property allows you to get the last SQL query that was used to p
dbIncremSearch	String	Allows you to specify the name of the button on the current form wit value of the component. The property is necessary to implement insta
dbItemID	Integer	The identifier of the selected record in the component. The identifier
dbListFieldsNames	String	Contains header names for columns separated by commas.
dbPopupMenu	TPopupMenu	Provides access to the component's popup menu. More info.
sqlValue	String	Returns the id of the selected record in the component for use in SQ *example:* SQLExecute ('INSERT INTO tablename (fieldname) VALUES (
BorderStyle	TBorderStyle	Определяет наличие бордюра у компонента. Доступные значения
CanFocus	Boolean	It checks if the component can get input focus, which is usually nece property, or if the component is located on a parent component with example: if Form1.TreeView1.CanFocus then Form1.TreeView1.SetFoc
Canvas	TCanvas	A class that allows you to draw on a component. More info.
Cell[x,y]	TCell	Property to access additional properties of the specified component
Cells[x,y]	String	Property for accessing the text content of a cell (x - column, y - row).
ClientWidth	Integer	The width of the client part of the component (i.e. excluding borders
ClientHeight	Integer	The height of the client part of the component (i.e., excluding the bo
Color	TColor	The background color of the component. More info.
Columns	TNxColumns	Property to access additional column properties and methods. More
Columns[i]	TNxCustomColumn	Property to access the properties of a specified column. More info.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it pa
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and
Expanded [Index: Integer]	Boolean	Defines the state of the node (expanded or collapsed).
FixedCols	Integer	Sets the number of fixed columns that will not move when scrolling h
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
HeaderSize	Integer	Gets or sets size of columns headers.
Hint	String	Hint contains the text string that appears when the user moves the m
HorzScrollBar	TNxScrollBar	Property to access additional scrollbar properties and methods. Mo
LastAddedRow	Integer	Returns the index of the last added line. Read-only.
Name	String	The name of the component.
Options	TGridOptions	Additional component settings. More info.
Parent	TWinControl	The parent component on which this component is placed.
RowCount	Integer	Gets total number of rows.
RowSize	Integer	Gets or sets default size (height) of rows in grid.
Row[i]	TRow	Access to additional properties of a row by its index. More info.
RowVisible[i]	Boolean	Determines the visibility of the row by its index.
Selected[i]	Boolean	Gets or sets specified row's selected state.
SelectedRow	Integer	Gets or sets selected Row's Index.
SelectedColumn	Integer	Gets or sets Index of selected Column.
SelectedCount	Integer	Gets number of selected rows.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer mo

TabOrder	Integer	Indicates the position of the component in its parent's tab order. Tab
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop t
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
VertScrollBar	TNxScrollBar	Property to access additional scrollbar properties and methods. Mo
Visible	Boolean	Specifies whether the component appears onscreen.
VisibleRows	Integer	Gets number of visible rows.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component
Top	Integer	Specifies the vertical coordinate of the upper-left of a component re
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	Description
function **dbIndexToID** (index: Integer): integer	Allows you to get the record ID (dat
function **dbUpdate**: String	Forcibly updates data in the compor was used to access the database.
function **AddRow** (Count: Integer = 1): Integer	Adds the specified number of rows
procedure **AddChildRow** (const Index: Integer; Position: TChildRowPosition)	Adds a child branch for the specifie
procedure **BeginUpdate**	Called before performing a large nu
procedure **BestFitColumns** (BestFitMode: TBestFitMode = bfCells)	The method automatically adjusts th
procedure **BestFitRow** (const Index: Integer)	The method automatically adjusts th
procedure **CalculateFooter** (VisibleOnly: Boolean = False)	Calculates footer of the component.
procedure **ClearRows**	Clears rows of of the component.
procedure **CollapseAll**	It collapses all the nodes of the com
procedure **DeleteRecord** (id: integer)	Deletes the record with the specifie
procedure **DeleteRow** (Index: Integer)	Deletes specified row. The record fr
function **ExportToExcel** (FileName: string = ''; ExcelVisible: boolean = True; FirsRowColumns: boolean = True): Variant	Exports data from the component t
function **ExportToLibreCalc** (FirsRowColumns: boolean = True): Boolean	Exports data from the component t
procedure **EndUpdate**	See BeginUpdate
procedure **ExpandNode** (Index: Integer)	Expands the specified node of the c
procedure **ExpandAll**	Expands all the nodes of the compo
function **GetRowAtPos** (X, Y: Integer): Integer	Allows you to get the row index by c
function **GetColumnAtPos** (X, Y: Integer): TNxCustomColumn	Allows you to get a link to a column
function **GetChildCount** (const Index: Integer; Recurse: Boolean = True): Integer	Returns the number of child records
function **GetFirstChild** (const Index: Integer): Integer	Returns the index of the first child r
function **GetLastChild** (const Index: Integer): Integer	Returns the index of the last child ro
function **GetLevel** (const Index: Integer): Integer	Returns the nesting level of the spec
function **GetNextSibling** (const Index: Integer): Integer	Returns the index of the next row or
function **GetParent** (const Index: Integer): Integer	Returns the index of the row that is t
function **GetPrevSibling** (const Index: Integer): Integer	Returns the index of the previous ro
function **HasChildren** (const Index: Integer): Boolean	Allows you to find out if there are ch
procedure **InsertRow** (Pos: Integer; Count: Integer = 1)	Inserts single row at specified positi
procedure **LoadFromTextFile** (const FileName: String; Separator: Char = ',';	Loads text file data into the compon

MultiLineSeparator: Char = '|'; StartRow: Integer = 0)	
procedure **MoveRow** (FromPos, ToPos: Integer)	Moves row from specified position t
procedure **SaveToHtml** (FileName: String; SaveHeaders: boolean = True; AllRows: boolean = False; CreateStyleSheet: boolean = True; SaveFooter: boolean = False; SaveCaption: boolean = False)	Saves the content of the component
procedure **SaveToTextFile** (const FileName: String; Separator: Char = ','; MultiLineSeparator: Char = '|')	Saves the contents of the componer
procedure **ScrollToRow** (index: integer)	Moves the scroll in the component s
procedure **SelectAll**	Selects all rows in the component. Fo
procedure **SelectRange** (FromRow, ToRow: Integer; Value: Boolean)	Selects (or deselects, if Value = False
procedure **SetFocus**	Gives the input focus to the compon
procedure **SwapRows** (FromPos, ToPos: Integer)	Swaps positions of two rows.

Events

Event	**Description**
OnAfterSort (Sender: TObject; ACol: Integer)	Occurs after the column is so
OnCellClick (Sender: TObject; ACol, ARow: Integer)	Occurs when the user clicks c
OnCellDoubleClick (Sender: TObject; ACol, ARow: Integer)	Occurs when the user double
OnChange (Sender: TObject)	Occurs after the component
OnClick (Sender: TObject)	Occurs when the user clicks t
OnColumnResize (Sender: TObject; ACol: Integer)	Occurs when the user resize
OnDoubleClick (Sender: TObject)	Occurs when the user double
OnEnter (Sender: TObject)	Occurs when a component re
OnExit (Sender: TObject)	Occurs when the input focus
OnExpand (Sender: TObject; ARow: Integer)	Occurs when a node expands
OnFooterClick (Sender: TObject; ACol: Integer)	Occurs when the user clicks c
OnHeaderClick (Sender: TObject; ACol: Integer)	Occurs when the user clicks c
OnHeaderDoubleClick (Sender: TObject; ACol: Integer)	Occurs when the user double
OnInputSelectCell (Sender: TObject; ACol: Integer)	Occurs when the user has mo
OnKeyDown (Sender: TObject; var Key: Word; Shift, Alt, Ctrl: boolean)	Occurs when a user presses
OnKeyPress (Sender: TObject; var Key: Char)	Occurs when a key is pressed
OnKeyUp (Sender: TObject; var Key: Word; Shift, Alt, Ctrl: boolean)	Occurs when the user release
OnLoadProgress (Sender: TObject; ACol, ARow: Integer)	Occurs when a text file is loa
OnMouseDown (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)	Occurs when the user presse
OnMouseEnter (Sender: TObject)	Occurs when the user moves
OnMouseLeave (Sender: TObject)	Occurs when the user moves
OnMouseMove (Sender: TObject; Shift, Alt, Ctrl: boolean; X, Y: Integer)	Occurs when the user moves
OnMouseUp (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)	Occurs when the user release
OnResize (Sender: TObject)	Occurs when a component is
OnRowMove (Sender: TObject; FromPos, ToPos: Integer; var Accept: Boolean)	Occurs when the user has mo
OnSortColumn (Sender: TObject; ACol: Integer; Ascending: Boolean)	Occurs before the column is
OnDropFiles (Sender: TObject; ArrayOfFiles: array of string; X, Y: Integer)	Occurs when user tries to dra

property dbPopupMenu: TPopupMenu

Description

Provides access to the component's popup menu.

Allows you to configure the popup menu.

Examples

```
// hide menu item
Form1.TreeView1.dbPopupMenu.Items[0].Visible := False;


// disable menu item
Form1.TreeView1.dbPopupMenu.Items[0].Enabled := False;


// programmatically click on the first menu item (numbering starts from 0)
Form1.TreeView1.dbPopupMenu.Items[0].Click;



// renaming menu items
procedure Form1_OnShow (Sender: TObject; Action: string);
begin
    Form1.TreeView1.dbPopupMenu.Items[0].Caption := 'Show record 2';
    Form1.TreeView1.dbPopupMenu.Items[1].Caption := 'Delete record 2';
    Form1.TreeView1.dbPopupMenu.Items[3].Caption := 'Copy cell 2';
    Form1.TreeView1.dbPopupMenu.Items[4].Caption := 'Copy 2';
    Form1.TreeView1.dbPopupMenu.Items[5].Caption := 'Copy all 2';
    Form1.TreeView1.dbPopupMenu.Items[7].Caption := 'Find 2';
end;



// adding menu items and submenu
procedure Form1_OnShow (Sender: TObject; Action: string);
var
   SubMenu: TMenuItem;
   MenuItem: TMenuItem;
begin
   SubMenu := TMenuItem.Create (Form1);
   SubMenu.Caption := 'SubMenu';
   MenuItem := TMenuItem.Create (Form1);
   MenuItem.Caption := 'Item';
   MenuItem.OnClick := @MenuClick1;
   Form1.TreeView1.dbPopupMenu.Items.Insert(0, SubMenu);
   Form1.TreeView1.dbPopupMenu.Items[0].Add(MenuItem);
end;
```

```
procedure MenuClick1;
begin
    ShowMessage('Hello from PopupMenu');
end;
```

property Cell[x,y]: TCell

Description

Property to access additional properties of the specified component cell (x - column, y - row).

The class has the following properties

Property	Description
AsBoolean: Boolean	Gets or sets value of item in Boolean type.
AsDateTime: TDateTime	Gets or sets value of item in TDateTime type.
AsFloat: Double	Gets or sets value of item in Double type.
AsInteger: Integer	Gets or sets value of item in Integer type.
Color: TColor	Gets or sets Cell Color.
Empty: Boolean	Determines whether the cell contains any value.
Hint: string	Gets or sets Cell Hint.
ObjectReference: TObject	Gets or sets Reference to TObject object.
FontStyle: TFontStyles	Gets or sets Cell Font Style. Available values: fsBold+fsItalic+fsUnderline+fsStrikeOut
Tag: Integer	Allows you to assign a number for your own needs.
TextColor: TColor	Specifies the font color for the cell.

Example

```
// changes the color of the first cell
procedure Form1_TreeView1_OnChange (Sender: TObject);
begin
    Form1.TreeView1.Cell[0,0].TextColor := clRed;
end;
```

property Columns: TNxColumns

Description

Property for accessing additional column properties and methods.

The class has the following properties

Properties and methods	Description
procedure **Add** (Source: TNxCustomColumn): TNxCustomColumn	Adds a new column with the specified class.

procedure **Clear**	Destroys all columns.
property **Count**: Integer	Returns the number of columns.
procedure **Delete** (index: integer)	Deletes the column with the specified index, the co
function **InsertCheckBoxColumn**(Pos: Integer): TNxCustomColumn	Inserting a column with a CheckBox (TNxCheckBox
function **InsertGraphicColumn**(Pos: Integer): TNxCustomColumn	Inserting a graphical column (TNxGraphicColumn)
function **InsertTreeColumn**(Pos: Integer): TNxCustomColumn	Inserting a tree column (TNxTreeColumn) in the sp
property **LastAdded**: TNxCustomColumn	Returns a reference to the last added column.

Examples

```
// As a rule, the creation of additional columns should happen in the event of
the OnChange component

// adding a column of the specified class, available classes:
// TNxTextColumn, TNxNumberColumn, TNxDateColumn, TNxTimeColumn,
TNxCheckBoxColumn, TNxListColumn, TNxGraphicColumn, TNxTreeColumn
try
   Form1.TreeView1.Columns.Add(TNxTextColumn);
except
end;
Form1.TreeView1.Columns.LastAdded.Color := clWhite;



// Inserting a column with a CheckBox (TNxCheckBoxColumn) at the specified
position, column numbering starts from 0.
Form1.TreeView1.Columns.InsertCheckBoxColumn(0);
```

property Columns[i]: TNxCustomColumn

Description

Property to access the properties of the specified column.

The class has the following properties

Property	**Type**	**Description**
Alignment	TAlignment	Sets the alignment of the column content. Available values: taCenter, ta
Enabled	Boolean	Determines the availability of the column to the user.
Footer	TColumnFooter	The property is responsible for the footer of the component. More inf
Header	TColumnHeader	The property is responsible for the headers of the component. More i
Options	TColumnOptions	Additional settings for the column. More info.
SlideAnchors	TAnchors	Makes sense when the GridStyles component property = gsSlides. Exa
SlideBounds	TNxSlideBounds	Makes sense when the GridStyles component property = gsSlides. Exa
SlideCaption	String	Makes sense when the GridStyles component property = gsSlides. Exa
SlideCaptionLocation	TSlideCatpionLocation	Makes sense when the GridStyles component property = gsSlides. Exa
Sorted	Boolean	Sorts the column.
SortKind	TSortKind	Specifies whether the column is sorted in ascending or descending orc

SortType	TSortType	Defines the sort type for the column. Values: stAlphabetic, stBoolean, s
Tag	Integer	Allows you to assign a number to the column for your own needs.
TagString	String	Allows you to assign a string to a column for your own needs.
VerticalAlignment	TVerticalAlignment	Sets the vertical alignment of the column content. Available values: vaT
Visible	Boolean	Determines the visibility of the column in the component.
Width	Integer	Sets the width of the column in the component.
WrapKind	TWrapKind	Sets the behavior of text that does not fit in the cell for a given column

property Footer: TColumnFooter

Description

This property is responsible for the footer of the component.

The class has the following properties

Property	Type	Description
Alignment	TAlignment	Sets the alignment of the footer content in the column. Available values: ta(
Caption	String	Footer text.
Color	TColor	Background Color.
FormulaKind	TFormulaKind	Formula for the calculation. Available values: fkNone, fkAverage, fkCount, fk
FormatMask	String	Allows you to set the format for numbers, text or date/time.
FormatMaskKind	TFormatMaskKind	Defines the way the mask in the FormatMask property is handled. Available
TextAfter	String	Specifies the text before the calculated value.
TextBefore	String	Specifies the text after the calculated value.

Example

```
// format the value in the basement for the columns of the numeric type (REAL,
CURRENCY, INTEGER)
// more details about using formatting
http://docwiki.embarcadero.com/Libraries/XE3/en/System.SysUtils.FormatFloat
procedure Form1_TableGrid1_OnChange (Sender: TObject);
begin
    TNxNumberColumn(Form1.TreeView1.Columns[0]).Footer.TextBefore := Price: ';
    TNxNumberColumn(Form1.TreeView1.Columns[0]).Footer.TextBefore := '$';
    TNxNumberColumn(Form1.TreeView1.Columns[0]).Footer.FormatMaskKind :=
mkFloat;
    TNxNumberColumn(Form1.TreeView1.Columns[0]).Footer.FormatMask :=
'#,##0.00';
end;
```

property Header: TColumnHeader

Description

This property is responsible for the footer of the component.

The class has the following properties

Property	Type	Description
Alignment	TAlignment	Sets the alignment of the header in the column. Available values: taCenter,
Caption	String	Title text.
Color	TColor	Header background color. Ignored if component property EnableVisualSt
DisplayMode	TDisplayMode	Defines the mode of displaying text and graphics in the header. Available
Glyph	TBitmap	Allows you to put a bmp image in the header.
Hint	String	A tooltip for the header.
MultiLine	Boolean	Allows you to use multiple strings in the header.
Orientation	THeaderOrientation	Defines the orientation of the header. Available values: hoHorizontal, hoVe

Example

```
// place the picture in the header
procedure Form1_TreeView1_OnChange (Sender: TObject);
begin
    Form1.TreeView1.Columns[0].Header.DisplayMode := dmTextAndImage;
    // get the image from Image1 placed on the form, the image must be in bmp
format
    Form1.TreeView1.Columns[
0].Header.Glyph.Assign(Form1.Image1.Picture.Bitmap);
end;
```

property Options: TColumnOptions

Description

Additional settings for the column.

The set has the following values.

Value	Description
coAutoSize	Not used. For automatic column widths, use the BestFitColumns method.
coCanClick	Determines whether the header of a given column can be clicked to sort.
coCanInput	Makes sense if the TableGrid.Options property has the value goInput. Allows you to enter a value
coCanSort	Determines whether sorting is executed when you click on the header of a given column.
coDisableMoving	Disables drag-and-drop of columns.
coEditing	Specifies the ability to edit the text in the rows of a given column.
coEditorAutoSelect	Determines whether text is automatically selected when the input cursor is set in a cell of a given

coFixedSize	Disables the resizing of this column.
coShowTextFitHint	Determines whether a hint will be shown if the contents of the cell do not fit.

Example

```
// As a rule, it is necessary to change column settings in the OnChange event
procedure Form1_TreeView1_OnChange (Sender: TObject);
begin
    Form1.TreeView1.Columns[0].Options := Form1.TreeView1.Columns[0].Options -
coCanSort; // disables sorting for the first column
    Form1.TreeView1.Columns[1].Options := Form1.TreeView1.Columns[1].Options +
coFixedSize; // Disables the ability to resize the second column
end;
```

property HorzScrollBar: TNxScrollBar

Description

A property for accessing additional scroll properties and methods.

The class has the following properties and methods

Properties and methods	**Description**
property **AutoHide**: Boolean	Determines whether to hide the scrollbar if it is not needed.
procedure **BeginUpdate**	The method disables the scroll update. After the necessary changes, you need t
procedure **EndUpdate**	The method enables a scroll update.
property **Enabled**: Boolean	Determines the availability of the scroll to the user.
procedure **First**	The method sets the scroll position to the beginning.
function **IsFirst**: Boolean	The function returns True if the scroll is at the beginning position.
function **IsLast**: Boolean	The function returns True if the scroll is at the end position.
procedure **Last**	The method sets the scroll position to the end.
property **ManualScroll**: Boolean	Allows you to enable manual scrolling. In this case, the scrolling will need to be i
property **MoveBy**(Distance: Integer)	The method moves the scroll position by the specified value. To move the scroll
procedure **Next**	The method moves the scroll position to the right (move step can be set in pixe
property **PageSize**: Integer	Returns the number of visible rows.
procedure **PageDown**	The method moves the scroll position, just like when you press the PageDown k
procedure **PageUp**	The method moves the scroll position, just like when you press the PageUp keyb
property **Position**: Integer	The property allows you to get or set a scroll position.
procedure **Prior**	The method moves the scroll position to the left (move step can be set in pixels
property **SmallChange**: Integer	Allows you to set the step for moving the scroll in pixels.
property **Visible**: Boolean	Determines the visibility of the scroll bar.

property VertScrollBar: TNxScrollBar

Description

A property for accessing additional scroll properties and methods.

The class has the following properties and methods

Properties and methods	**Description**
property **AutoHide**: Boolean	Determines whether to hide the scrollbar if it is not needed.
procedure **BeginUpdate**	The method disables the scroll update. After the necessary changes, you need t
procedure **EndUpdate**	The method enables a scroll update.
property **Enabled**: Boolean	Determines the availability of the scroll to the user.
procedure **First**	The method sets the scroll position to the beginning.
function **IsFirst**: Boolean	The function returns True if the scroll is at the beginning position.
function **IsLast**: Boolean	The function returns True if the scroll is at the end position.
procedure **Last**	The method sets the scroll position to the end.
property **ManualScroll**: Boolean	Allows you to enable manual scrolling. In this case, the scrolling will need to be i
property **MoveBy**(Distance: Integer)	The method moves the scroll position by the specified value. To move the scroll
procedure **Next**	The method moves the scroll position to the down (move step can be set in pixe
property **PageSize**: Integer	Returns the number of visible rows.
procedure **PageDown**	The method moves the scroll position, just like when you press the PageDown k
procedure **PageUp**	The method moves the scroll position, just like when you press the PageUp keyb
property **Position**: Integer	The property allows you to get or set a scroll position.
procedure **Prior**	The method moves the scroll position to the up (move step can be set in pixels
property **SmallChange**: Integer	Allows you to set the step for moving the scroll in pixels.
property **Visible**: Boolean	Determines the visibility of the scroll bar.

property Options: TGridOptions

Description

Additional component settings.

You can combine the settings using the listed values.

Value	**Description**
goArrowKeyExitEditing	Exit cell editing by pressing up, down, left, right buttons
goCanHideColumn	The user will be able to hide columns with the mouse.
goDisableColumnMoving	Disables the user to move columns with the mouse.
goDisableKeys	Disables moving between rows with arrows on the keyboard.
goEscClearEdit	When editing a cell, allows you to clear it by pressing Esc.
goFooter	Makes the component's footer visible.
goGrid	Shows horizontal and vertical lines.
goHeader	Shows the headings for the columns.
goIndicator	Shows the indicator of the selected row.

goInput	Shows the input bar.
goLockFixedCols	Disables moving fixed columns with mouse (FixedCols).
goMultiSelect	Allows you to select multiple records using the Ctrl or Shift key.
goRowResizing	Allows you to change the height of rows with the mouse (goIndicator must also be activate
goRowMoving	Allows you to move rows with the mouse.
goSecondClickEdit	Editing a cell with a double click of the mouse.
goSelectFullRow	Selects a whole row, otherwise individual cells will be selected.

The default settings are as follows:
goDisableColumnMoving, goHeader, goIndicator, goSecondClickEdit, goSelectFullRow

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
begin
    // disable lines in the component
    Form1.TreeView1.Options := Form1.TreeView1.Options - goGrid;

    // Set the necessary settings for the component
    Form1.TreeView1.Options := goDisableColumnMoving + goGrid + goHeader +
goSecondClickEdit + goSelectFullRow + goMultiSelect;
end;
```

property Row[i]: TRow

Description

Property to access additional properties of the specified row.

The class has the following properties

Property	Description
ChildCount: Integer	Number of child elements (makes sense if there is a TNxTreeColumn).
Expanded: Boolean	Whether the row is expanded to show the children (makes sense if there is a TNxTreeColu
HasChildren: Boolean	Does the row have children (makes sense if there is a TNxTreeColumn).
ID: Integer	Contains record ID (id field from database).
ImageIndex: Integer	Icon index from TImageList in property TNxTreeColumn(Form1.TreeView1.Columns[0]).Ima
Level: Integer	Contains the nesting level (makes sense if there is a TNxTreeColumn).
ParentRow: TRow	Reference to the parent row (makes sense if there is a TNxTreeColumn).
RowHeight: Integer	Gets or sets the height of a Row.
Selected: Boolean	Allows you to know if a row is selected or not.
Shown: Boolean	Allows you to know if the given row is visible (makes sense if there is a TNxTreeColumn).

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
```

```
begin
   // show the ID of the specified row
   ShowMessage( Form1.TreeView1.Row[0].ID );
end;
```

procedure BestFitColumns(BestFitMode: TBestFitMode = bfCells)

Description

The method allows you to automatically adjust the width of the columns.

As a rule, this method should be called in the event of the OnChange component.

You can specify the following values as a parameter:

bfCells - autosize width by cells contents.
bfBoth - autosize width by contents of cells and column titles.
bfHeader - autosize width by column headers.

Example

```
procedure Form1_TreeView1_OnChange (Sender: TObject);
begin
    Form1.TreeView1.BestFitColumns(bfBoth);
end;
```

procedure BestFitRow(const Index: Integer)

Description

The method automatically adjusts the height of the row, depending on the contents of the cells in this row.

As a rule, this method must be called in the event of the OnChange component.
Also, you must assign the WrapKind := wkWordWrap property to columns to allow wrap text in cells.

Example

```
// automatically adjust the height of all rows in the component
procedure Form1_TreeView1_OnChange (Sender: string);
var
    i, c: integer;
begin
    c := Form1.TreeView1.Columns.Count - 1;
    for i := 0 to c do
    begin
        Form1.TreeView1.Columns[i].VerticalAlignment := taAlignTop; // set the
vertical alignment in the cells of the column
```

```
        Form1.TreeView1.Columns[i].WrapKind := wkWordWrap; // enable the
possibility of wrapping the strings in the cells of the column
    end;

    c := Form1.TreeView1.RowCount - 1;
    for i := 0 to c do Form1.TreeView1.BestFitRow(i); // for each row we call
the method for auto height adjustment
end;


// also updates the height of the rows when you resize the columns
procedure Form1_TreeView1_OnColumnResize (Sender: TObject; ACol: Integer);
var
    i, c: integer;
begin
    c := Form1.TreeView1.RowCount - 1;
    for i := 0 to c do Form1.TreeView1.BestFitRow(i); // for each row we call
the method for auto height adjustment
end;
```

procedure OnCellClick(Sender: TObject; ACol, ARow: Integer)

Description

It is triggered when the user clicked on a cell.

The event contains parameters ACol and ARow, which contain the column number and row number, respectively, on which the user clicked.

Example

```
// show the contents of the cell clicked on by the user
procedure Form1_TableGrid1_OnCellClick (Sender: TObject; ACol, ARow: Integer);
begin
    ShowMessage( Form1.TreeView1.Cells[ACol, ARow] );
end;
```

Map

Description

Allows you to place an interactive geographical map of Google Maps on the form, with the ability to put on the map markers, lines and polygons (placing lines and polygons is done using scripts.).

Class: TdbMap

Properties

Property	Type	Description
APIKey	String	Optionally specify an API Key to identify the application with the Google Maps A
APIChannel	String	Optionally specify a Channel ID to identify the application with the Google Maps
APIClientID	String	Optionally specify a Client ID to identify the application with the Google Maps P
APISignature	String	Optionally specify an API Signature to identify the application with the Google M
APIClientAuthURL	String	Optionally specify the authenticated URL as specified on the Google Maps Prem specified.
CurrentLocation	TLocation	Contains the coordinates of the user's location. For these coordinates to be ava
dbTable	String	Determines which database table a component belongs to.
dbFieldLatitude	String	Specifies in which field of the database table the latitude value of the marker wi
dbFieldLongitude	String	Specifies in which field of the database table the longitude value of the marker v
DisableMenu	Boolean	Allows you to disable the map context menu.
Elevations	TElevations	Contains the result of calling the GetElevation and GetElevation2 method. Allows
FormMarkerName	String	Specifies the name of the form that will be used when creating/editing the mark
LastAddedMarker	TMarker	Contains a reference to the last added marker using the method: TMarkers.Add
LastAddedPolyline	TPolylineItem	Contains a reference to the last added path using the method: TPolylines.Add (F
LastAddedPolygon	TPolygonItem	Contains a reference to the last polygon added using the method: TPolygons.Ac
Markers	TMarkers	Property for working with markers on the map. More info.
Markers[i]	TMarker	Property to access existing markers on the map. More info.
MapOptions	TMapOptions	Setting up the component. More info.
Polylines	TPolylines	Property for working with polylines on the map. More info.
Polylines[i]	TPolylineItem	Property to access existing polylines on the map. More info.
Polygons	TPolygons	Property for working with polygons on the map. More info.
Polygons[i]	TPolygonItem	Property to access existing polygons on the map. More info.
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer ever
Name	String	The name of the component.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relative to i
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative to its
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.

Methods

Method	Description
function **AddGeoImage** (FileName: string; Title: string = ''; Width: Integer = -1; Height: Integer = -1; ZoomWidth: Integer = -1; ZoomHeight: Integer = -1): TMarker	This function retrieves the geo coordir a Marker to the map at that location w parameters to specify a custom size fo
function **AddMapKMLLayer** (Url: string; ZoomToBounds: Boolean): Boolean	This function displays a KML file on th bounding box of the contents of the la
function **ClearPolygons**: Boolean	Remove all polygons from the map ar
function **ClearPolylines**: Boolean	Remove all polylines from the map an
function **CloseMarkerInfoWindowHtml** (Id: Integer): Boolean	The function closes the information wi
procedure **CloseAllMarkersInfoWindow**	Closes the information windows for all
function **CreateMapPolygon** (Polygon: TMapPolygon): Boolean	Puts the specified polygon on the map
function **CreateMapPolyline** (Polyline: TPolyline): Boolean	Puts the specified polyline on the map

function **DegreesToLonLat** (StrLon, StrLat: String; var Lon,Lat: Double): Boolean	This function converts degrees to long
function **DeleteAllMapKMLLayer**: Boolean	This function removes all KML layers fr
function **DeleteMapKMLLayer** (Id: Integer): Boolean	The function removes a KML layer fror
function **DeleteMapMarker** (Id: Integer): Boolean	Deletes the specified marker from the
function **DeleteMapPolyline** (Id: Integer): Boolean	Deletes the specified polyline from the
function **DeleteMapPolygon** (Id: Integer): Boolean	Deletes the specified polygon from th
function **Distance** (la1, lo1, la2, lo2: Double): Double	Calculates the distance between two c
function **Focused**: Boolean	Determines whether the control has in
function **GetCurrentLocation**: Boolean	Sets the CurrentLocation.Latitude and
function **GetMapBounds**: Boolean	This function retrieves the bounds coo Doesn't work in the current version.
function **GetElevation** (Latitude, Longitude: Double): Boolean	Retrieves the elevation data for a sing the request succeeded, false otherwise
function **GetElevation2** (Path: TPath; ResultCount: Integer = 2): Boolean	Retrieves the elevation data for a Path end location (last coordinate in path) a intervals. The number of intervals is ind the request succeeded, false otherwise
function **GetModifiedMapPolyline** (Polyline: TPolyline):Boolean	This function retrieves modified coord modifies lines on the map with the mou
function **GetModifiedMapPolygon** (Polygon: TMapPolygon):Boolean	This function retrieves modified coord for a Polygon of type ptPath, the Cente ptRectangle) Typically used when the u
function **GetPolygonAreaSqMeters** (APolygonId: Integer): string	Returns the area in square meters for
function **LoadGeoJSONPolyline** (AFilename: string; AColor: TColor = clBlue; Opacity: Integer = 255; AWidth: Integer = 2; Zoom: Boolean = True; HoverColor: TColor = clBlue): string	This function loads coordinates from a Opacity, Width, HoverColor of the Poly More info.
function **LoadGeoJSONPolygon** (AFilename: string; BorderColor: TColor = clBlue; Opacity: Integer = 255; BackgroundColor: TColor = clBlue; BackgroundOpacity: Integer = 100; AWidth: Integer = 2; Zoom: Boolean = True; HoverBorderColor: TColor = clBlue; HoverBackgroundColor: TColor = clBlue): string	This function loads coordinates from a BorderColor, Opacity, BackgroundColo Optionally set Zoom to true to automa
function **LoadGPSRoute** (AFilename: string; AColor: TColor = clRed; AWidth: integer = 2; ZoomToRoute: Boolean = False): string	This function loads a GPS route from a
procedure **LoadMarkersFromPoi** (PoiFile: string; MarkerColor: TMarkerIconColor = icDefault)	This functions loads a set of coordinat Optionally the color of the markers car
function **LonLatToXY** (Lon, Lat: Double; var X, Y: Integer): Boolean	This function converts longitude / latitu the latitude and longitude coordinates.
function **MapPanTo** (Latitude,Longitude:Double): Boolean	This function performs a pan to a loca the control canvas.
function **MapZoomTo** (Bounds: TBounds): Boolean	This function performs a zoom to fit th
function **MapPanBy** (X,Y: Integer): Boolean	The function moves the map horizonta
function **OpenMarkerInfoWindowHtml** (Id: Integer; HtmlText:String): Boolean	The function opens the marker info wi HtmlText string. More info.
procedure **SaveMarkersToPoi** (PoiFile: string)	This functions saves the coordinates o
function **ScreenShot** (ImgType: TImgType): TGraphic	The function takes a screenshot of the More info.
procedure **SetFocus**	Gives the input focus to the componer
procedure **SwitchToStreetView**	Switches the map to the panoramic str
procedure **SwitchToMap**	Exit the panoramic street view mode.
function **UpdateMapMarkers**: Boolean	Updates all markers on the map.
function **UpdateMapMarker** (Marker: TMarker):Boolean	Updates the specified marker on the r
function **UpdateMapPolygon** (Polygon: TMapPolygon): Boolean	Updates the specified polygon on the script.

function **UpdateMapPolyline** (Polyline: TPolyline): Boolean	Updates the specified polyline on the script.
function **XYToLonLat** (X, Y: integer; var Lon, Lat: double): Boolean	This function converts XY coordinates pixel coordinates in the control window

Events

Event	Description
OnMapClick (Sender: TObject; Latitude, Longitude: Double; X, Y: Integer; MouseLeft, MouseRight, MouseMiddle: Boolean)	Occurs when the map is clicked. Retu coordinates in the control window, bu
OnMapDblClick (Sender: TObject; Latitude, Longitude: Double; X, Y: Integer)	Occurs when the user double-clicks t longitude coordinates of the mouse c window.
OnMapMouseEnter (Sender: TObject; Latitude, Longitude: Double; X, Y: Integer)	Occurs when the user moves the mou the X and Y values indicate the pixel p
OnMapMouseExit (Sender: TObject; Latitude, Longitude: Double; X, Y: Integer)	Occurs when the user moves the mou position, the X and Y values indicate t
OnMapMouseMove (Sender: TObject; Latitude, Longitude: Double; X, Y: Integer)	Occurs when the user moves the mou coordinates of the mouse cursor pos
OnMapMove (Sender: TObject; Latitude, Longitude: Double; X, Y: Integer)	Occurs when the entire map is move mouse cursor position, the X and Y va
OnMapMoveEnd (Sender: TObject; Latitude, Longitude: Double; X, Y: Integer)	Occurs at the end of an entire map n position, the X and Y values indicate t
OnMapMoveStart (Sender: TObject; Latitude, Longitude: Double; X, Y: Integer)	Occurs at the start of an entire map r position, the X and Y values indicate t
OnMapTypeChange (Sender: TObject; NewMapType: TMapType)	Occurs when the map type is change
OnMapZoomChange (Sender: TObject; NewLevel: Integer)	Occurs when the zoom level is chang
OnMarkerClick (Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double; MouseLeft, MouseRight, MouseMiddle: Boolean)	Occurs when a marker is clicked. Retu what button has been clicked on the r
OnMarkerDblClick (Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double)	Occurs when a marker is double-click selected marker.
OnMarkerDrag (Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double)	Occurs when a marker is dragged ar coordinates of the selected marker.
OnMarkerDragEnd (Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double)	Occurs at the end of when a marker coordinates of the selected marker.
OnMarkerDragStart (Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double)	Occurs at the start of when a marker coordinates of the selected marker.
OnMarkerInfoWindowCloseClick (Sender: TObject; IdMarker: Integer)	Occurs when the info window is close
OnMarkerMouseDown (Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double)	Occurs when the mouse cursor is ove and longitude coordinates of the sele
OnMarkerMouseEnter (Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double)	Occurs when the mouse cursor enter position.
OnMarkerMouseExit (Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double)	Occurs when the mouse cursor leave values indicate the pixel position in th
OnMarkerMouseUp (Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double)	Occurs when the mouse cursor is ove and longitude coordinates of the sele
OnAfterAddMarker (Sender: TObject; IdMarker: Integer)	Occurs after adding a marker. The ev
OnBeforeDeleteMarker (Sender: TObject; IdMarker: Integer; var Cancel: Boolean)	Occurs before the marker is deleted,

	script.
OnPolygonClick (Sender: TObject; IdPolygon: Integer; MouseLeft, MouseRight, MouseMiddle: Boolean)	Occurs when a polygon is clicked. Re
OnPolygonDblClick (Sender: TObject; IdPolygon: Integer)	Occurs when a polygon is double-clic
OnPolygonMouseDown (Sender: TObject; IdPolygon: Integer)	Occurs when the mouse cursor is ove
OnPolygonMouseEnter (Sender: TObject; IdPolygon: Integer)	Occurs when the mouse cursor enter
OnPolygonMouseExit (Sender: TObject; IdPolygon: Integer)	Occurs when the mouse cursor leave
OnPolygonMouseUp (Sender: TObject; IdPolygon: Integer)	Occurs when the mouse cursor is ove
OnPolygonChanged (Sender: TObject; IdPolygon: Integer)	Occurs when a polygon has been mo with the GetModifiedPoygon method.
OnPolylineClick (Sender: TObject; IdPolyline: Integer; MouseLeft, MouseRight, MouseMiddle: Boolean)	Occurs when a polyline is clicked. Ret
OnPolylineDblClick (Sender: TObject; IdPolyline: Integer)	Occurs when a polyline is double-clic
OnPolylineMouseDown (Sender: TObject; IdPolyline: Integer)	Occurs when the mouse cursor is ove
OnPolylineMouseEnter (Sender: TObject; IdPolyline: Integer)	Occurs when the mouse cursor enter
OnPolylineMouseExit (Sender: TObject; IdPolyline: Integer)	Occurs when the mouse cursor leave
OnPolylineMouseUp (Sender: TObject; IdPolyline: Integer)	Occurs when the mouse cursor is ove
OnPolylineChanged (Sender: TObject; IdPolyline: Integer)	Occurs when a polyline has been mo the GetModifiedPoyline method.
OnStreetViewChange (Sender: TObject; Heading, Pitch, Zoom: Integer)	Occurs when the Point Of View is cha
OnStreetViewMove (Sender: TObject; Latitude, Longitude: Double; X, Y: Integer)	Occurs when the geographic positior position, the X and Y values indicate t
OnWebGMapsError (Sender: TObject; ErrorType: TErrorType)	Occurs when an error is received. Th

type TMarkerIconColor

Description

Type to define the color of the marker.

Available values:

Value	Color
icDefault	Default
icBlue	Blue
icGreen	Green
icRedicPurple	Purple

type TErrorType

Description

Type for determining an error when working with a component.

Available values:

Value
etGMapsProblem

etScreenshotProblem
etJavascriptError
etNotValidMarker
etStreetViewUnknownError
etStreetViewNoResults
etInvalidWaypoint

class TBounds

Description

The class is designed to indicate the northeast and southwest coordinates. The class is used to gets or sets a rectangular area on the map.

Class Properties

Property
NorthEast.Latitude: Double
NorthEast.Longitude: Double
SouthWest.Latitude: Double
SouthWest.Longitude: Double

Example

```
// showing Paris on the map
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    Bounds: TBounds;
begin
    Bounds := TBounds.Create;
    Bounds.NorthEast.Latitude := 48.900868;
    Bounds.NorthEast.Longitude := 2.396142;
    Bounds.SouthWest.Latitude := 48.817004;
    Bounds.SouthWest.Longitude := 2.244114;
    Form1.Map1.MapZoomTo(Bounds);
    Bounds.Free;
end;
```

property Markers: TMakers

Description

Property for working with markers on the map. Allows you to create/modify/delete markers on the map.

Methods and properties	**Description**
function **Add**(Latitude, Longitude: Double): TMarker	Creates a new marker on the map.
function **Bounds**: TBounds	Returns the coordinates of the rectangular area in which all mar
procedure **Clear**	Removes all markers on the map and from the collection (TMar
property **Count**: Integer	Returns the number of markers in the collection (TMarkers).
procedure **Delete**(Index: Integer)	Removes the specified marker from the map and from the colle
property **Items[i]**: TMarker	Returns the specified marker.

Example

```
//Create a marker on the map
var
  Marker: TMarker;
  i: integer;
  id: integer;
begin
  // Create a marker on the map with specified coordinates
  Marker := Form1.Map1.Markers.Add(48.858137, 2.294476);



  // Set the map zoom to fit all existing markers
  Form1.Map1.MapZoomTo(Form1.Map1.Markers.Bounds);
```

property Markers[i]: TMarker

Description

Property to access existing markers on the map. Allows you to change/delete markers on the map.

Property	**Type**	**Description**
Clickable	Boolean	When set to true, enables clicking on the marker. Clicking opens an extra info
Data	String	Store extra data associated with the marker.
dbID	Integer	Allows to find out the marker id in the database (sense when more than 1 mar
Draggable	Boolean	When set to true, the marker can be moved around the map when dragged.
Icon	String	Allows the use of an image as a marker. A local path to an image file or an url format: File://C:/folder/iconname.png
IconColor	TMarkerIconColor	Allows changing the color of the default marker icon to one of the available pr
IconWidth	Integer	Specify a custom width value in pixels for the marker icon. Can only be used w displayed in its full size.
IconHeight	Integer	Specify a custom height value in pixels for the marker icon. Can only be used v

		displayed in its full size.
Index	Integer	Allows you to get the index of a marker on the map.
Latitude	Double	Sets the latitude value of the marker on the map.
Longitude	Double	Sets the longitude value of the marker on the map.
MapLabel	TMapLabel	Allows the use of a HTML label displayed on top of the marker. The label is au
Tag	Integer	Allows you to assign a number to a component for your own needs.
Text	String	The text that will be displayed over the marker.
Title	String	Text that will be visible as a tooltip for the marker.
Visible	Boolean	Determines the visibility of the marker on the map.

Example

```
//display the coordinates of all markers on the map
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    i, c: integer;
    lat, lng: Double;
begin
    c := Form1.Map1.Markers.Count-1;
    for i := 0 to c do
    begin
        lat := Form1.Map1.Markers[i].Latitude;
        lng := Form1.Map1.Markers[i].Longitude;
        ShowMessage(Coordinates: ' + FloatToStr(lat)+', '+ FloatToStr(lng));
    end;
end;
```

property MapLabel: TMapLabel

Description

The class is designed to customize the label for the marker.

Class Properties

Property	Type	Description
BorderColor	TColor	The border color of the label.
Color	TColor	The color of the label.
Font	TFont	The font for the label text.
Margin	Integer	The margin in pixels between the label border and the label text.
OffsetLeft	Integer	The left offset of the label relative to the marker coordinates. This is a percentage value. Fo example the value 0 will center align the label, the value 50 will right align the label.
OffsetTop	Integer	The top offset of the label relative to the marker coordinates. This is a pixel value. With a d Font.Size and the default Marker the label is displayed on top of the Marker.
Text	String	The text displayed in the label. If this value is empty, no label is displayed. You can use htm

property Polylines: TPolylines

Description

Property for working with polylines on the map. Allows you to create/modify/delete polylines on the map.

Methods and properties	Description
function **Add**: TPolylineItem	Creates a new polyline object.
function **Bounds**: TBounds	Returns the coordinates of the rectangular area in which all the polylines are locate
procedure **Clear**	Removes all polylines from the collection. It will not remove polylines from the map ClearPolylines method to remove polylines simultaneously from the map and TPolyl
property **Count**: Integer	Returns the number of polylines in the collection (TPolylines).
procedure **Delete**(Index: Integer)	Deletes the specified polyline from the collection (TPolylines). To remove a polyline DeleteMapPolyline method.
property **Items[i]**: TPolylineItem	Returns the specified polyline.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    PolylineItem: TPolylineItem;
begin
    PolylineItem := Form1.Map1.Polylines.Add; // Create an object for a
polyline
    PolylineItem.Polyline.Width := 2;
    PolylineItem.Polyline.Path.Add(50, 2); // add a starting point
    PolylineItem.Polyline.Path.Add(52, 4); // add a line
    PolylineItem.Polyline.Path.Add(50, 4); // add a line

    Form1.Map1.CreateMapPolyline(PolylineItem.Polyline); // display the
created polyline on the map
    Form1.Map1.MapZoomTo(PolylineItem.Polyline.PathBounds); // set the map
zoom to fit the created polyline
end;
```

property Polylines[i]: TPolylineItem

Description

Property for accessing existing polylines on the map. Allows you to change/delete polylines on the map.

Свойства	Описание
property **Clickable**: Boolean	When set to true, enables clicking on the polyline. If the value is False, then the OnP
property **Color**: TColor	The color of the polyline.
property **Editable**: Boolean	When set to true, the polyline can be edited.

property **Geodesic**: Boolean	When set to true, each edge is rendered as a geodesic. When set to false, render e
property **HoverColor**: TColor	The color of the polyline when hovered.
property **Opacity**: Integer	The opacity of the polyline. (values 1-100).
property **Path**: TPath	The ordered sequence of coordinates of the polyline.
property **Path[i]**: TPathItem	Returns a point with coordinates by its index.
property **PathBounds**: TBounds	Returns the coordinates of the rectangular area where the given polyline is located.
property **Tag**: Integer	Allows you to assign a number to a component for your own needs.
property **TagString**: string	Allows you to assign a string to a component for your own needs.
property **TagObject**: TObject	The object associated with the polyline
property **Width**: Integer	The width of the polyline in pixels.
property **Visible**: Boolean	When set to true, the polyline is shown on the map.
property **Zindex**: Integer	The zIndex compared to other elements on the map.

Example

```
// move all polylines by 0.001 latitude and longitude
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    iLine, iPath, cLine, cPath: integer;
begin
    cLine := Form1.Map1.Polylines.Count-1;
    for iLine := 0 to cLine do
    begin
        cPath := Form1.Map1.Polylines[iLine].Polyline.Path.Count-1;
        for iPath := 0 to cPath do
        begin
            Form1.Map1.Polylines[iLine].Polyline.Path[iPath].Latitude :=
Form1.Map1.Polylines[iLine].Polyline.Path[iPath].Latitude + 0.001;
            Form1.Map1.Polylines[iLine].Polyline.Path[iPath].Longitude :=
Form1.Map1.Polylines[iLine].Polyline.Path[iPath].Longitude + 0.001;
        end;
        Form1.Map1.UpdateMapPolyline(Form1.Map1.Polylines[iLine].Polyline); //
update the polyline on the map
    end;
end;
```

property Path: TPath

Description

Property for working with individual points that make up a polyline. Allows you to create/modify/delete polyline points.

Methods and properties	Description
function **Add** (Latitude, Longitude: double): TPathItem	Adds a new point for the polyline.

procedure **Clear**	Deletes all points in the given polyline.
property **Count**: Integer	Returns the number of points in the given polyline.
procedure **Delete**(Index: Integer)	Deletes the specified point on the given polyline.
property **Items[i]**: TPathItem	Returns the specified point in the given polyline.

property Path[i]: TPathItem

Description

Contains the geographical coordinates of the point.

Property	**Description**
property **Latitude**: Double	Широта точки.
property **Longitude**: Double	Долгота точки.

property Polygons: TPolygons

Description

Property for working with polygons on the map. Allows you to create/modify/delete polygons on the map.

Methods and properties	**Description**
function **Add**: TPolygonItem	Creates a new polygon object.
procedure **Bounds**: TBounds	Contains the coordinates of the rectangular area in which all polygons are located.
procedure **Clear**	Removes all polygons from the collection. The polygons from the map will not be the map and TPolygons collection.
property **Count**: Integer	Returns the number of polygons in the collection (TPolygons).
procedure **Delete**(Index: Integer)	Removes the specified polygon from the collection (TPolygons). At that, the polygo DeleteMapPolygon method.
property **Items[i]**: TPolygonItem	Returns the specified polygon.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    PolygonItem: TPolygonItem;
```

```
begin
    // Circle
    PolygonItem := Form1.Map1.Polygons.Add;
    PolygonItem.Polygon.BackgroundOpacity := 50;
    PolygonItem.Polygon.BorderWidth := 2;
    // Setting up a polygon with the Circle type
    PolygonItem.Polygon.PolygonType := ptCircle;
    PolygonItem.Polygon.Radius := 10000;
    PolygonItem.Polygon.Center.Latitude := 50;
    PolygonItem.Polygon.Center.Longitude := 2;
    Form1.Map1.CreateMapPolygon(PolygonItem.Polygon);

    // Rectangle
    PolygonItem := Form1.Map1.Polygons.Add;
    PolygonItem.Polygon.BackgroundOpacity := 50;
    PolygonItem.Polygon.BorderWidth := 2;
    //Setting up a polygon with Rectangle type
    PolygonItem.Polygon.PolygonType := ptRectangle;
    PolygonItem.Polygon.Bounds.NorthEast.Latitude := 52;
    PolygonItem.Polygon.Bounds.NorthEast.Longitude := 4;
    PolygonItem.Polygon.Bounds.SouthWest.Latitude := 50;
    PolygonItem.Polygon.Bounds.SouthWest.Longitude := 3;
    Form1.Map1.CreateMapPolygon(PolygonItem.Polygon);

    // Polygons are based on a list of longitude and latitude coordinates
    PolygonItem := Form1.Map1.Polygons.Add;
    PolygonItem.Polygon.BackgroundOpacity := 50;
    PolygonItem.Polygon.BorderWidth := 2;
    //Setting up a polygon
    PolygonItem.Polygon.PolygonType := ptPath;
    PolygonItem.Polygon.Path.Add(50, 2);
    PolygonItem.Polygon.Path.Add(52, 4);
    PolygonItem.Polygon.Path.Add(50, 4);
    Form1.Map1.CreateMapPolygon(PolygonItem.Polygon);

    Form1.Map1.MapZoomTo(Form1.Map1.Polygons.Bounds); // Set the map zoom to
fit all existing polygons
end;
```

property Polygons[i]: TPolygonItem

Description

Property for accessing existing polygons on the map. Allows you to change/delete polygons on the map.

Methods and properties	Описание
property **BackgroundColor**: TColor	The color of the polygon.
property **BackgroundOpacity**: Integer	The opacity of the polygon. (values: 1-100).
property **BorderColor**: TColor	The border color of the polygon.
property **BorderOpacity**: Integer	The border opacity of the polygon. (values: 1-100).
property **BorderWidth**: Integer	The width of the polygon border in pixels.
property **Bounds**: TBounds	Sets the bounds of a polygon when PolygonType is set to ptRectangle.
property **Center**: TLocation	Sets the latitude/longitude of the center point of the circle when PolygonTy
property **Clickable**: Boolean	When set to true, enables clicking on the polygon. If the value is False, ther

property **Editable**: Boolean	When set to true, the polygon can be edited.
property **Geodesic**: Boolean	When set to true, each edge is rendered as a geodesic. When set to false,
property **HoverBackgroundColor**: TColor	The color of the polygon when hovered.
property **HoverBorderColor**: TColor	The border color of the polygon when hovered.
property **Path**: TPath	The ordered sequence of coordinates of the polygon that forms a closed
property **Path[i]**: TPathItem	Returns the coordinates of a point by its index. Makes sense if the polygo
property **PathBounds**: TBounds	Returns the coordinates of the rectangular area in which this polygon is lo
property **PolygonType**: TPolygonType	Sets the type of polygon to be rendered. Available values: ptCircle, ptRect
property **Radius**: Integer	The radius of the polygon in meters. (When PolygonType is set to ptCircle)
property **Tag**: Integer	Allows you to assign a number to a component for your own needs.
property **TagString**: string	The text associated with the polygon (optional). The appearance of the hin is set to true, this value will be displayed as a hint when hovering the polyg
property **TagObject**: TObject	Allows you to assign an object to a polygon for your own use.
property **Visible**: Boolean	When set to true, the polygon is shown on the map.
property **Zindex**: Integer	The zIndex compared to other elements on the map.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    PolygonItem: TPolygonItem;
begin
    // Circle
    PolygonItem := Form1.Map1.Polygons.Add;
    PolygonItem.Polygon.BackgroundOpacity := 50;
    PolygonItem.Polygon.BorderWidth := 2;
    // Setting up a polygon with the Circle type
    PolygonItem.Polygon.PolygonType := ptCircle;
    PolygonItem.Polygon.Radius := 10000;
    PolygonItem.Polygon.Center.Latitude := 50;
    PolygonItem.Polygon.Center.Longitude := 2;
    Form1.Map1.CreateMapPolygon(PolygonItem.Polygon);

    // Rectangle
    PolygonItem := Form1.Map1.Polygons.Add;
    PolygonItem.Polygon.BackgroundOpacity := 50;
    PolygonItem.Polygon.BorderWidth := 2;
    //Setting up a polygon with Rectangle type
    PolygonItem.Polygon.PolygonType := ptRectangle;
    PolygonItem.Polygon.Bounds.NorthEast.Latitude := 52;
    PolygonItem.Polygon.Bounds.NorthEast.Longitude := 4;
    PolygonItem.Polygon.Bounds.SouthWest.Latitude := 50;
    PolygonItem.Polygon.Bounds.SouthWest.Longitude := 3;
    Form1.Map1.CreateMapPolygon(PolygonItem.Polygon);

    // Polygons are based on a list of longitude and latitude coordinates
    PolygonItem := Form1.Map1.Polygons.Add;
    PolygonItem.Polygon.BackgroundOpacity := 50;
    PolygonItem.Polygon.BorderWidth := 2;
    //Setting up a polygon
    PolygonItem.Polygon.PolygonType := ptPath;
    PolygonItem.Polygon.Path.Add(50, 2);
    PolygonItem.Polygon.Path.Add(52, 4);
    PolygonItem.Polygon.Path.Add(50, 4);
    Form1.Map1.CreateMapPolygon(PolygonItem.Polygon);

    Form1.Map1.MapZoomTo(Form1.Map1.Polygons.Bounds); // Set the map zoom to
fit all existing polygons
end;
```

property MapOptions: TMapOptions

Description

Setting up the component. The class has the following properties

Property	**Description**
property **DefaultToCurrentLocation**: Boolean	Sets the current location as the default position when Launch is called.
property **DefaultLatitude**: Double	Sets the latitude value for the default position when Launch is called.
property **DefaultLongitude**: Double	Sets the longitude value for the default position when Launch is called
property **DisableDoubleClickZoom**: Boolean	When set to true, disables zoom functions when double-clicking.
property **DisablePOI**: Boolean	When set to true, disable display of the points of interest on the map.
property **DisableTilt**: Boolean	Disable the auto-tilted view on satellite maptype. Note: tilted view is o the Google Maps API.
property **Draggable**: Boolean	When set to true, the entire map can be moved around in the control.
property **EnableKeyboard**: Boolean	When set to true, enables the use of the keyboard for controlling pan
property **Language**: TLanguageName	Defines the language of the map. Available values: lnDefault, lnArabic, lnEnglish, lnEnglish_Australian, lnEnglish_GreatBritain, lnSpanish, lnFarsi, lnHungarian, lnIndonesian, lnItalian, lnHebrew, lnJapanese, lnKannada, lnPolish, lnPortuguese, lnPortuguese_Brazil, lnPortuguese_Portugal, lnR lnTamil, lnTelugu, lnThai, lnTurkish, lnUkrainian, lnVietnamese, lnChinese
property **MapType**: TMapType	Sets the type of map. Available values: mtDefault, mtSatellite, mtHybri
property **ShowTraffic**: Boolean	When set to true, and if available in your country, traffic information c
property **ShowBicycling**: Boolean	When set to true, and if available in your country, bicycle trail informa
property **ScrollWheel**: Boolean	When set to true, enables the use of the scroll wheel. The scroll wheel
property **ZoomMap**: TZoomMap	Is to be used to set the default zoom at startup. The zoom value is a v

function GetElevation (Latitude, Longitude: Double): Boolean

Description

Retrieves the elevation data for a single latitude and longitude coordinate. The data is added to the Elevations collection. Returns true if the request succeeded, false otherwise. Maps Elevation API activation is required.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
begin
    if Form1.Map1.GetElevation(49, 2) then
    begin
        if Form1.Map1.Elevations.Count > 0 then
ShowMessage(Form1.Map1.Elevations[0].Elevation) else ShowMessage('No result');
    end;
end;
```

function GetElevation2 (Path: TPath; ResultCount: Integer = 2): Boolean

Description

Retrieves the elevation data for a Path that contains latitude and longitude coordinates. The start location (first coordinate in Path) and end location (last coordinate in path) are used to form a straight line. The elevation data is retrieved along the straight path at specified intervals. The number of intervals is indicated by the ResultCount parameter. The data is added to the Elevations collection. Returns true if the request succeeded, false otherwise. Maps Elevation API activation is required.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
    i, c: integer;
begin
    if Form1.Map1.GetElevation2(Form1.Map1.Polylines[0].Polyline.Path) then
    begin
        c := Form1.Map1.Elevations.Count-1;
        for i := 0 to c do
        begin
            ShowMessage(FloatToStr(Form1.Map1.Elevations[i].Elevation));
        end;
    end else ShowMessage('Error');
end;
```

function DegreesToLonLat (StrLon, StrLat: String; var Lon, Lat: Double): Boolean

Description

This function converts degrees to longitude / latitude coordinates. The result of the function is contained in the variables Lon and Lat.

Example

```
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
   Lon, Lat: Double;
begin
   if Form1.Map1.DegreesToLonLat('17°47''19.809"E', '49°31''46.604"N', Lon,
Lat) then
   begin
      ShowMessage('Longitude: ' + FloatToStr(Lon));
      ShowMessage('Latitude: ' + FloatToStr(Lat));
   end;
end;
```

function LoadGeoJSONPolyline (AFilename: string; AColor: TColor = clBlue; Opacity: Integer = 255; AWidth: Integer = 2; Zoom: Boolean = True; HoverColor: TColor = clBlue): string

Description

This function loads coordinates from a GEOJSON file and displays it on the map as a Polyline or Polylines. Optionally set the Color, Opacity, Width, HoverColor of the Polyline(s). Optionally set Zoom to true to automatically zoom the map to the bounds of the Polyline(s).

Function parameters:

Parameter	Type	Description
AFilename	String	GeoJSON file name.
AColor	TColor	The color that will be used to draw the polyline. The default color is blue. Optional paran
Opacity	Integer	The degree of transparency of the polyline. Values from 0 to 255. Optional parameter.
AWidth	Integer	The width of the polyline. The default width is two pixels. Optional parameter.
Zoom	Boolean	Determine whether the polyline should be automatically shown on the map. Optional par
HoverColor	TColor	The color the polyline will have when the mouse cursor is over it. The default color is blu

An example of a GeoJSON file:

```
{
    "type":"FeatureCollection","features":[
        {
            "type":"Feature",
            "properties": {
                "name": "Polyline1"
            },
            "geometry":{
                "type": "LineString",
                "coordinates": [[
                    [-105.431410315776, 20.878495854271],
                    [-105.445432904506, 20.8727217105441],
                    [-105.451183560633, 20.8762102822492]
                ]]
            }
        },


        {
            "type":"Feature",
            "properties": {
                "name":"Polyline2"
            },

            "geometry":{
                "type": "LineString",
                "coordinates": [[
                    [-105.491410315776, 20.878495854271],
                    [-105.495432904506, 20.8727217105441],
                    [-105.491183560633, 20.8762102822492]
                ]]
            }
        }


    ]
}
```

Read more about the GeoJSON format here https://geojson.org/

Example

```
Form1.Map1.LoadGeoJSONPolyline('D:\polylines.geojson');
```

function LoadGeoJSONPolygon

function **LoadGeoJSONPolygon** (AFilename: string; BorderColor: TColor = clBlue; Opacity: Integer = 255; BackgroundColor: TColor = clBlue; BackgroundOpacity: Integer = 100; AWidth: Integer = 2; Zoom: Boolean = True; HoverBorderColor: TColor = clBlue; HoverBackgroundColor: TColor = clBlue): string

Description

This function loads coordinates from a GEOJSON file and displays it on the map as a Polygon or Polygons. Optionally set the BorderColor, Opacity, BackgroundColor, BackgroundOpacity, Width, HoverBorderColor, HoverBackgroundColor of the Polygon(s). Optionally set Zoom to true to automatically zoom the map to the bounds of the Polygon(s).

Function parameters:

Parameter	Type	Description
AFilename	String	GeoJSON file name.
BorderColor	TColor	The color that will be used to border the polygons. The default color is blue. Option
Opacity	Integer	The degree of transparency of the polygon border. Values from 0 to 255. Optional p
BackgroundColor	TColor	The color that will be used to build the polygon. The default color is blue. Optional p
BackgroundOpacity	Integer	The degree of transparency of the polygons. Values from 1 to 100. Optional parame
AWidth	Integer	The width of the border for polygons. The default width is two pixels. Optional para
Zoom	Boolean	Determine whether the polyline should be automatically shown on the map. Optiona
HoverBorderColor	TColor	The color that the border of the polygon will have when the mouse cursor is over it.
HoverBackgroundColor	TColor	The color the polygon will have when the mouse cursor is over it. The default color i

An example of a GeoJSON file:

```
{
    "type":"FeatureCollection","features":[
        {
            "type":"Feature",
            "properties": {
                "name":"Polygon1"
            },

            "geometry":{
                "type":"Polygon",
                "coordinates": [[
                    [-105.431410315776, 20.878495854271],
                    [-105.445432904506, 20.8727217105441],
                    [-105.451183560633, 20.8762102822492]
                ]]
            }
        },


        {
            "type":"Feature",
            "properties": {
                "name":"Polygon2"
            },

            "geometry":{
                "type":"Polygon",
                "coordinates": [[
                    [-105.491410315776, 20.878495854271],
                    [-105.495432904506, 20.8727217105441],
                    [-105.491183560633, 20.8762102822492]
                ]]
            }
        }


    ]
}
```

Подробней о GeoJSON формате можно прочитать здесь https://geojson.org/

Example

```
Form1.Map1.LoadGeoJSONPolygon('D:\polygons.geojson');
```

function OpenMarkerInfoWindowHtml (Id: Integer; HtmlText:String): Boolean

Description

The function opens the marker info window for the marker with selected marker-id (Marker.Index). Extra information can be passed via the HtmlText string.

Example

```
procedure Form1_Map1_OnMarkerClick (Sender: TObject; MarkerTitle: string;
IdMarker: Integer; Latitude, Longitude: Double; MouseLeft, MouseRight,
MouseMiddle: boolean);
begin
    Form1.Map1.CloseAllMarkersInfoWindow; // closes the previous info windows
    Form1.Map1.OpenMarkerInfoWindowHtml(IdMarker,'<b>'+ MarkerTitle + '<br>' +
'Lat : ' + floattostr(latitude)+ '<br>' + 'Lon : ' + floattostr(longitude) +
'</b>');
end;
```

function ScreenShot (ImgType: TImgType): TGraphic

Description

The function takes a screenshot of the actual map canvas. This screenshot is taken in the chosen imagetype: itJPeg, itBitmap or itPng. The graphic can easily be saved to file.

Values
itBitmap
itJpeg
itPng

Examples

```
// Save the map image to a file
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
   png: TPngImage;
begin
   png := TPngImage(Form1.Map1.ScreenShot(itPng));
   png.SaveToFile('d:\filename.png');
   png.Free;
end;


// transfer the map image to Image in Bitmap format
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
var
   bmp: TBitmap;
begin
   bmp := TBitmap(Form1.Map1.ScreenShot(itBitmap));
   Form1.Image1.Picture.Bitmap.Assign(bmp);
   bmp.Free;
end;
```

Image

Description

Use Image to display a graphical image on a form. For example, you can place your company logo on the form. Supported formats are jpg, bmp, gif, png8, png24.

Class: TdbImage

Properties

Property	Type	Description
AutoSize	Boolean	Specifies whether the control sizes itself automatically to accommodate its contents.
Canvas	TCanvas	Provides a drawing surface for embellishing bitmap images. More info.
Center	Boolean	Indicates whether the image is centered in the image control.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes into the reg
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer events.
Hint	String	Hint contains the text string that appears when the user moves the mouse over the c
Name	String	The name of the component.
Picture	TPicture	Specifies the image that appears on the image component. More info.
Proportional	Boolean	Indicates whether the image should be changed, without distortion, so that it fits the
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves over the cor
Stretch	Boolean	Indicates whether the image should be changed so that it exactly fits the bounds of t
Tag	Integer	Allows you to assign a number to a component for your own needs.

TagString	String	Allows you to assign a string to a component for your own needs.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relative to its par
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative to its pare
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	Description
procedure **Clear**	Clears the picture of the component.
function **CanPasteBitmapFromClipboard**: boolean	The function returns True if the clipboard contains an image in bm
function **CopyToClipboard**: boolean	The function copies the component's picture to the clipboard. If su
function **PasteBitmapFromClipboard**: boolean	The function inserts an image from the clipboard into the compon

Events

Event	Description
OnClick (Sender: TObject)	Occurs when the user clicks the component.
OnDoubleClick (Sender: TObject)	Occurs when the user double-clicks the left m
OnMouseDown (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)	Occurs when the user presses a mouse butto
OnMouseEnter (Sender: TObject)	Occurs when the user moves the mouse into a
OnMouseLeave (Sender: TObject)	Occurs when the user moves the mouse outsi
OnMouseMove (Sender: TObject; Shift, Alt, Ctrl: boolean; X, Y: Integer)	Occurs when the user moves the mouse point
OnMouseUp (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)	Occurs when the user releases a mouse butto

property Picture: TPicture

Description

TPicture contains a bitmap, icon or metafile graphic. If the TPicture contains a bitmap graphic, the Bitmap property specifies the graphic. If the TPicture contains an icon graphic, the Icon property specifies the graphic. If the TPicture contains a metafile graphic, the Metafile property specifies the graphic.

Methods and properties	Description
procedure **Assign** (Source: TPersistent)	Copies one object to another by copying the contents of that object `Form1.Image2.Picture.Assign(Form1.Image1.Picture);`
property **Bitmap**: TBitmap	Specifies the contents of the picture object as a bitmap graphic (.BM

property Bitmap.Canvas: TCanvas	Provides access to a drawing surface that represents the bitmap. Mo
property **Graphic**: TGraphic	Specifies the graphic that the picture contains.
property **Height**: Integer	Specifies the vertical size (in pixels) of the graphic. Read only.
procedure **LoadFromFile** (const Filename: string)	Reads the file specified in Filename and loads the data into the TPictu
procedure **SaveToFile** (const Filename: string)	Writes the picture to disk.
property **Width**: Integer	Specifies the horizontal size (in pixels) of the picture. Read only.

PageControl

Description

PageControl is a set of pages used to make a multiple page dialog box.

Class: TdbPageControl

Properties

Property	**Type**	**Description**
ActivePage	TTabSheet	Specifies the page currently displayed by the page control.
ActivePageIndex	Integer	Specifies the page currently displayed by the page control. Use ActivePageIndex to in the Pages property array. Changing the value of ActivePageIndex changes the Acti Setting ActivePageIndex to a value that is out of bounds (less than 0 or greater than Unlike the TabIndex property, ActivePageIndex returns the index of the selected tab,
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary before us component is located on a parent component with those properties, using the SetFo example: if Form1.PageControl1.CanFocus then Form1.PageControl1.SetFocus;
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes into the re
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer events.
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
Hint	String	Hint contains the text string that appears when the user moves the mouse over the
MultiLine	Boolean	Determines whether the tabs can appear on more than one row.
Name	String	The name of the component.
Pages[i]	TTabSheet	Allows you to access the properties of the desired tab by its index.
PageCount	Integer	Indicates the number of pages in the PageControl component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves over the co
Style	TTabStyle	Specifies the style of the tab control. Available values: tsTabs (default), tsButtons (bu

TabIndex	Integer	Identifies the selected tab on a tab control. If there is no selected tab, it returns -1. Unlike the ActivePageIndex property, TabIndex returns the sequence number of the s False).
TabPosition	TTabPosition	Determines whether tabs appear at the top or bottom. Available values: tpTop, tpBo
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrder is the o
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allow or disal
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relative to its pa
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative to its par
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	**Description**
procedure **SetFocus**	Gives the input focus to the component.

Events

Event
OnChange (Sender: TObject)
OnChanging (Sender: TObject; var AllowChange: Boolean)
OnEnter (Sender: TObject)
OnExit (Sender: TObject)
OnMouseDown (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnMouseEnter (Sender: TObject)
OnMouseLeave (Sender: TObject)
OnMouseMove (Sender: TObject; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnMouseUp (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnResize (Sender: TObject)
OnDropFiles (Sender: TObject; ArrayOfFiles: array of string; X, Y: Integer)

class TTabSheet

Description

TabSheet is an individual page in a PageControl component.

Allows you to control the tabs of the PageControl component.

Class: TdbTabSheet

Properties

Property	Description
BorderWidth: Integer	Specifies the width of the control's border.
Caption: string	Defines the title of the tab.
ControlCount: Integer	Returns the number of child controls.
Controls[i]: TControl	Access to a control by its index.
Cursor: TCursor	Specifies the image used to represent the mouse pointer when it passes into t
Enabled: Boolean	Controls whether the component responds to mouse, keyboard, and timer eve
Font: TFont	Allows you to set font name, size, color and style. More info.
Hint: string	Hint contains the text string that appears when the user moves the mouse ove
Name: string	The name of the component.
PageIndex: Integer	Indicates the index of the tab sheet in the list of tab sheets maintained by the p
ShowHint: Boolean	ShowHint specifies whether to show the Help Hint when the mouse pointer mo
TabIndex: Integer	Indicates the position of the tab sheet in the set of visible tabs in a PageContro false, the TabIndex property is -1.
TabVisible: Boolean	Specifies whether the tab of the TTabSheet object appears in its PageControl.
Tag: Integer	Allows you to assign a number to a component for your own needs.
TagString: string	Allows you to assign a string to a component for your own needs.
Visible: Boolean	Specifies whether the component appears onscreen.
Left: Integer	Specifies the horizontal coordinate of the left edge of a component relative to
Top: Integer	Specifies the vertical coordinate of the upper-left of a component relative to it
Width: Integer	Specifies the horizontal size of the component in pixels.
Height: Integer	Specifies the vertical size of the component in pixels.

Methods

Метод	Описание
function **CanFocus**: Boolean	It checks if the component can get input focus, which is usually necessary before us False then using the SetFocus method will cause an error. Example: `if Form1.PageControl1.Pages[0].CanFocus then Form1.Pag`
function **Focused**: boolean	Determines whether the control has input focus.
procedure **SetFocus**	Gives the input focus to the component.

События компонента

Событие
OnEnter (Sender: TObject)
OnExit (Sender: TObject)

OnMouseDown (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnMouseEnter (Sender: TObject)
OnMouseLeave (Sender: TObject)
OnMouseMove (Sender: TObject; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnMouseUp (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnResize (Sender: TObject)
OnDropFiles (Sender: TObject; ArrayOfFiles: array of string; X, Y: Integer)

Example

```
// Change the title of the first tab
procedure Form1_Button1_OnClick (Sender: TObject; var Cancel: boolean);
begin
   Form1.PageControl1.Pages[0].Caption := 'Tab name';
end;
```

GroupBox

Description

The GroupBox component represents a standard Windows group box, used to group related controls on a form.

Class: TdbGroupBox

Properties

Property	Type	Description
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary before us component is located on a parent component with those properties, using the SetFo example: if Form1.GroupBox1.CanFocus then Form1.GroupBox1.SetFocus;
Caption		Defines the title of the component.
Color	TColor	Specifies the background color of the control. The color change for this component
ControlCount	Integer	Returns the number of child controls.
Controls[i]	TControl	Access to a control by its index.

Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes into the re
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer events.
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. The color change for this compor
Hint	String	Hint contains the text string that appears when the user moves the mouse over the
Name	String	The name of the component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves over the co
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrder is the o
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allow or disal
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relative to its pa
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative to its par
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	**Description**
procedure **SetFocus**	Gives the input focus to the component.

Events

Event
OnClick (Sender: TObject)
OnDoubleClick (Sender: TObject)
OnEnter (Sender: TObject)
OnExit (Sender: TObject)
OnMouseDown (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnMouseEnter (Sender: TObject)
OnMouseLeave (Sender: TObject)
OnMouseMove (Sender: TObject; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnMouseUp (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnDropFiles (Sender: TObject; ArrayOfFiles: array of string; X, Y: Integer)

Panel

Description

Use Panel to put an empty panel on a form.

Class: TdbPanel

Properties

Property	Type	Description
AutoSize	Boolean	Specifies whether the component sizes itself automatically to accommodate its cont
BevelOuter	TBevelCut	Determines the style of the outer bevel of a panel. Available values: bvRaised, bvLov
CanFocus	Boolean	It checks if the component can get input focus, which is usually necessary before us component is located on a parent component with those properties, using the SetFo example: if Form1.Panel1.CanFocus then Form1.Panel1.SetFocus;
Canvas	TCanvas	Specifies the TCanvas object that presents a drawing surface for the component. M
Caption		The text that will be displayed in the center of the component.
Color	TColor	Specifies the background color of the control. The color change for this component
ControlCount	Integer	Returns the number of child controls.
Controls[i]	TControl	Access to a control by its index.
Cursor	TCursor	Specifies the image used to represent the mouse pointer when it passes into the re
Enabled	Boolean	Controls whether the component responds to mouse, keyboard, and timer events.
Focused	Boolean	Determines whether the control has input focus.
Font	TFont	Allows you to set font name, size, color and style. More info.
Hint	String	Hint contains the text string that appears when the user moves the mouse over the
Name	String	The name of the component.
ShowHint	Boolean	Specifies whether to show the Help Hint when the mouse pointer moves over the co
TabOrder	Integer	Indicates the position of the component in its parent's tab order. TabOrder is the o
TabStop	Boolean	Determines whether the user can tab to a control. Use the TabStop to allow or disal
Tag	Integer	Allows you to assign a number to a component for your own needs.
TagString	String	Allows you to assign a string to a component for your own needs.
Visible	Boolean	Specifies whether the component appears onscreen.
Left	Integer	Specifies the horizontal coordinate of the left edge of a component relative to its pa
Top	Integer	Specifies the vertical coordinate of the upper-left of a component relative to its par
Width	Integer	Specifies the horizontal size of the component in pixels.
Height	Integer	Specifies the vertical size of the component in pixels.

Methods

Method	Description
procedure **SetFocus**	Gives the input focus to the component.

Events

Event
OnClick (Sender: TObject)
OnDoubleClick (Sender: TObject)

OnEnter (Sender: TObject)
OnExit (Sender: TObject)
OnMouseDown (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnMouseEnter (Sender: TObject)
OnMouseLeave (Sender: TObject)
OnMouseMove (Sender: TObject; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnMouseUp (Sender: TObject; MouseLeft, MouseRight, MouseMiddle: boolean; Shift, Alt, Ctrl: boolean; X, Y: Integer)
OnResize (Sender: TObject)
OnDropFiles (Sender: TObject; ArrayOfFiles: array of string; X, Y: Integer)

OnDropFiles

Description

Occurs when an attempt is to drag a file from explorer to the component for which this event was created. It also allows you to get the names of the files that were dragged to the component by the user.

Example

```
procedure Form1_Memo1_OnDropFiles (Sender: TObject; ArrayOfFiles: array of
string; X, Y: Integer);
var
    i, c: integer;
begin
    c := Length(ArrayOfFiles)-1;
    for i := 0 to c do
    begin
        Form1.Memo1.Lines.Add(ArrayOfFiles[i]);
    end;
end;
```

Classes

TCanvas

Description

TCanvas provides properties and methods that assist in creating an image by:

- Specifying the type of brush, pen, and font to use.
- Drawing and filling a variety of shapes and lines.
- Writing text.

- Rendering graphic images.

Class Methods and Properties

Methods and Properties	Description
procedure **Arc** (X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer)	Use Arc to draw an elliptical the points (X1,Y1) and (X2,Y2 to the ending point. The start and (X3,Y3). The ending poin (X4, Y4).
procedure **ArcTo** (X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer)	Draws an arc on the image a perimeter of an ellipse that is ellipse, counterclockwise, from ellipse and a line defined by and a line defined by the cen
procedure **AngleArc** (X, Y: Integer; Radius: Cardinal; StartAngle, SweepAngle: Single)	Draws an arc on the image a from the current position to t traverses the perimeter of a perimeter of the circle, count If the sweep angle is greater
property **Brush**: TBrush	Determines the color and pa
property **Brush.Color**: TColor	Indicates the color of the bru
property **Brush.Style**: TBrushStyle	Specifies the pattern for the bsCross, bsDiagCross
procedure **Chord** (X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer)	Draws a closed figure repres an arc and a line that joins th (X1,Y1) and (X2,Y2). The ellips
procedure **CopyRect** (destLeft, destTop, destRight, destBottom: integer; Canvas: TCanvas; srcLeft, srcTop, srcRight, srcBottom: integer	Copies part of an image fro to the image of the TCanvas Canvas parameter specifies t canvas that will be copied.
procedure **Draw** (X, Y: Integer; Graphic: TGraphic)	Renders the graphic specifie
procedure **Draw2** (X, Y: Integer; Graphic: TGraphic; Opacity: Byte)	Renders the graphic specifie Opacity parameter allows yo
procedure **Ellipse** (X1, Y1, X2, Y2: Integer)	Draws the ellipse defined by bottom right point at (X2, Y2 Pen, and filled using the valu
property **Font**: TFont	The property is responsible
procedure **LineTo** (X, Y: Integer)	Draws a line on the canvas f
procedure **MoveTo** (X, Y: Integer)	Changes the current drawing
property **Pen**: TPen	Specifies the kind of pen the
property **Pen.Color**: TColor	Determines the color used t
property **Pen.Style**: TPenStyle	Determines the style in which psClear, psInsideFrame
property **Pen.Width**: Integer	Specifies the width of the pe
procedure **Pie** (X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer)	Draws a pie-shaped section
property **Pixels** (X, Y: Integer): TColor	Specifies the color of the pix pixel position within the curre returns -1. Write Pixels to cha

	image.
procedure **Rectangle** (X1, Y1, X2, Y2: Integer)	Draws a rectangle on the ca
procedure **RoundRect** (X1, Y1, X2, Y2, X3, Y3: Integer)	Draws a rectangle with roun Brush. The rectangle will have create a rounded appearanc Y3.
procedure **TextOut** (X, Y: Integer; const Text: string)	Writes a string on the canva be written using the current v
property **Handle**: Integer	Specifies the handle for this

Example

```
    Form1.Image1.Canvas.Brush.Style := bsClear;
    Form1.Image1.Canvas.Font.Orientation := 270;
    Form1.Image1.Canvas.MoveTo(50, 50);
    Form1.Image1.Canvas.LineTo(100, 100);
    Form1.Image1.Canvas.TextOut(150, 150, 'Texts');
    Form1.Image1.Canvas.Ellipse(30, 30, 45, 45);
```

TFont

Description

TFont describes font characteristics used when displaying text.

Properties

Property	Description
Color: TColor	Specifies the color of the text.
Name: String	Identifies the typeface of the font.
Size: Integer	Specifies the height of the font in points.
Style: TFontStyles	Determines whether the font is normal, italic, underlined, bold, and so on.

Example

```
Form1.Button1.Font.Color := clRed;
Form1.Button1.Font.Name := 'Arial';
Form1.Button1.Font.Size := 14;
Form1.Button1.Font.Style := fsBold + fsItalic + fsUnderline + fsStrikeOut;
```

TSizeConstraints

Description

Specifies the size constraints for the control. Use Constraints to specify the minimum and maximum width and height of the control. When Constraints contains maximum or minimum values, the component cannot be resized to violate those constraints.

Example

```
Form1.Constraints.MaxWidth := 800;
Form1.Constraints.MaxHeight := 600;
Form1.Constraints.MinWidth := 200;
Form1.Constraints.MinHeight := 150;
```

TStringList

Description

TStringsList introduces many properties and methods to:

- Add or delete strings at specified positions in the list.
- Rearrange the strings in the list.
- Access the string at a particular location.
- Read the strings from or write the strings to a file or stream.
- Associate an object with each string in the list.
- Store and retrieve strings as name-value pairs.

Properties

Свойство	**Назначение**
Count: Integer	The number of strings in the list.
Sorted: Boolean	Specifies whether the strings in the list should be automatically sorted.
Text: String	Lists the strings in the TStringsList object as a single string with the individual strings delimited by car

Methods

Метод	**Назначение**
function **Add** (const S: string): Integer	Adds a new string to the list. Add returns the position of the item
procedure **Clear**	Deletes all the strings from the list.
procedure **Delete** (Index: Integer)	Removes the string specified by the Index parameter.
function **Find** (s: string; var Index: integer): Boolean	Locates the index for a string in a sorted list and indicates whethe does not contain a string that matches S, Find returns false. Only u method instead.
function **IndexOf** (const S: string): Integer	Returns the position of a string in the list. If the string does not ha
procedure **Insert** (Index: Integer; const S: string)	Inserts a string to the list at the position specified by Index. If the Use Add with sorted lists.

procedure **LoadFromFile** (const FileName: string)	Fills the string list with the lines of text in a specified file.
procedure **Move** (CurIndex, NewIndex: Integer)	Changes the position of a string in the list. Use Move to move the NewIndex.
procedure **SaveToFile** (const FileName: string)	Saves the strings in the current object to the specified FileName fi
procedure **Sort**	Sorts the strings in the list in ascending order.

Example

```
var
   sl: TStringList;
begin
   sl := TStringList.Create;
   try
      sl.Add('String 1');
      sl.Add('String 2');
      sl.Add('String 3');
      sl.Insert(1, 'One more string'); // inserts a new string in the list,
the numbering starts from zero.
      sl.SaveToFile('d:\textfile.txt');
   finally
      sl.Free;
   end;
```

Types

TColor

Description

TColor is used to specify the color of a component.

You can use hexadecimal numbers to specify an arbitrary color.
example:

```
Form1.Color := $00DDEEFF;
```

where FF - red, EE - green, DD - blue.

If you are more accustomed to ordinary numbers, you can use the RGB function, where each color is

specified by a number from 0 to 255.
example:

```
Form1.Color := RGB(255, 238, 221);
```

In addition, you can use text color constants
example:

```
Form1.Color := clWindowText;
Form1.Color := clRed;
```

The following table lists the color constants:

Value	Meaning
clBlack	Black
clMaroon	Maroon
clGreen	Green
clOlive	Olive Green
clNavy	Navy Blue
clPurple	Purple
clTeal	Teal
clGray	Gray
clSilver	Silver
clRed	Red
clLime	Lime Green
clYellow	Yellow
clBlue	Blue
clFuchsia	Fuchsia
clAqua	Aqua
clWhite	White

The following table lists the colors that are defined in the Windows Control panel:

Value	Meaning
clActiveBorder	Current border color of the active window.
clActiveCaption	Current color of the active window's title bar.
clAppWorkSpace	Current color of the application workspace.
clBackground	Current background color of the Windows desktop.
clBtnFace	Current color of a button face.
clBtnHighlight	Current color of the highlighting on a button.
clBtnShadow	Current color of a shadow cast by a button.
clBtnText	Current color of text on a button.
clCaptionText	Current color of the text on the active window's title bar.
clGradientActiveCaption	Windows 98 or Windows 2000: Right-side color in the color gradient of an active window's ti clActiveCaption specifies the left side color.

clGradientInactiveCaption	Windows 98 or Windows 2000: Right-side color in the color gradient of an inactive window's clInactiveCaption specifies the left side color.
clGrayText	Current color of dimmed text.
clHighlight	Current background color of selected text.
clHighlightText	Current color of selected text.
clHotLight	
clInactiveBorder	Current border color of inactive windows.
clInactiveCaption	Current color of inactive windows' title bar.
clInactiveCaptionText	Current color of the text on an inactive window's title bar.
clInfoBk	Windows 95 or NT 4.0 only: Background color for tool tip controls.
clInfoText	Windows 95 or NT 4.0 only: Text color for tool tip controls.
clMenu	Current background color of menus.
clMenuBar	Current color of the menu bar.
clMenuHighlight	Current color of the highlighting on a menu.
clMenuText	Current color of text on menus.
clScrollBar	Current color of the scroll bar track.
cl3DDkShadow	Windows 95 or NT 4.0 only: Dark shadow for three-dimensional display elements.
cl3DLight	Windows 95 or NT 4.0 only: Light color for three-dimensional display elements (for edges fa light source).
clWindow	Current background color of windows.
clWindowFrame	Current color of window frames.
clWindowText	Current color of text in windows.

TCursor

Description

TCursor identifies the cursor type.

A variable of type TCursor can have one of the following values:

Value	Description
crAppStart	Hour glass and standard pointer combination cursor shown at application startup
crArrow	Default cursor.
crCross	Fine cross-shaped cursor used in graphic applications for precise positioning.
crDefault	Default cursor: thick arrow pointing up and left.
crDrag	Drag cursor for single items being dragged.
crHandPoint	An upward pointing hand cursor. This is normally used to identify a selectable item, such as a web pa
crHelp	Question mark and standard pointer operation.
crHourGlass	Hour glass cursor to indicate a busy process.
crHSplit	Cursor shown when the mouse is over a horizontal splitter.
crIBeam	Text insert cursor in the form of a thin capital I
crMultiDrag	Drag cursor for multiple items being dragged.
crNo	Black cross in a black circle indicating invalid mouse target location.
crNoDrop	White cross in a white circle indicating that a drag operation is hovering over an invalid drop target.
crSizeAll	Cursor for resizing up, down, left, and right.
crSizeNESW	Resizing cursor from North-West to South-East resizing.

crSizeNS	Vertical resizing cursor.
crSizeNWSE	Resizing cursor from North-West to South-East resizing.
crSizeWE	Horizontal resizing cursor.
crSQLWait	Hour glass cursor to indicate a busy SQL database operation.
crUpArrow	Thin upward pointing cursor.
crVSplit	Cursor shown when the mouse is over a vertical splitter.

Example

```
Form1.Cursor := crHandPoint;
```

TDateTime

Description

The TDateTime type holds a date and time value. It is stored as a Double variable, with the date as the integral part, and time as fractional part.
Because TDateTime is actually a double, you can perform calculations on it as if it were a number. This is useful for calculations such as the difference between two dates.

Examples

Components

Examples with the Map component

- Creating a marker on the map
- Show the specified area on the map
- Cycle through all markers on the map
- Creating polylines on the map
- Moving all polylines on the map
- Creating polygons on the map (circle, rectangle, polygon)
- Get the height of the Earth's surface for the specified coordinates
- Get the height of the ground surface for the specified path
- Converting text representation of geographic coordinates to latitude and longitude
- Creating polylines on the map based on data from GeoJSON file
- Creating polygons on the map based on data from GeoJSON file
- When you click on the marker, the label will show
- Take a screenshot from the map
- Adding a marker to a map with additional data

Examples with the Calendar component

Select the days on the calendar

Examples with the TableGrid component

- Populate a component with data based on an SQL query
- Adding and renaming popup menu items
- Automatic height of rows depending on their content
- How to make a timetable
- How to make a timetable 2
- Next and Previous buttons
- Gantt chart
- Striped TableGrid
- Change the color of the cell component TableGrid

Examples with the TreeView component

- Adding and renaming popup menu items
- Automatic height of rows depending on their content

Examples with the DBFile component

- How to add multiple files to the database

Examples with the DBImage component

- How to resize picture before save to database

Examples with the Button component

- Own icons for buttons

Examples with the Counter component

- Creating custom counter (eg .: MS-0001, MS-0002)

Database

- Automatic database backup (SQLite)

- Logging with using triggers (SQLite)
- Check the existence of record before saving

Files

- Searching for files on disk

Internet

- Sending E-mail message with a file
- MySQL with SSL
- How to connect to MySQL using script
- Sending SMS
- Reads email messages (POP3)

Report

- Send report document to E-mail
- Display an image in a report when using the LinkFile
- Subreports
- Print barcode
- The figures in words

Others

- How to translate system interface and messages
- COM port
- Create trial project
- How to work with CSV files
- Work with SNMP devices (exm: monitoring printers on the network)
- Windows Video Media Player
- Webcam integration using ffmpeg.exe utility
- Using .chm help file in your project
- How to make Installer for your project using InnoSetup
- Regular expression
- Creating your own menu item "About" and a window with information
- Hide the main menu (File, Options, About)
- Run project on Windows startup
- Calculating on the form