

Overview

Korzh Query Builder (KQB) is not just another one query building program.

It provides really easy-to-use interface for building queries to SQL databases: all query conditions are described in natural language as plain English terms.

This feature of KQB allows to create new queries even by the users who are not very experienced in relational databases and SQL language. They can build their own queries without needing to understand database design or table relationships.

During query building Simple Query shows long field descriptions instead of actual field names, operators spelled out (e.g. "is equal to") rather than mathematical symbols and field value descriptions (e.g. "MasterCard") rather than actual values ("MC").

KQB demonstrates the abilities provided by [Simple Query \(Active Query\) library](#). This library allows any developer to include his (her) own query builder into any application written in Borland Delphi or C++ Builder (Simple Query) or Visual Basic, Access, Visual C++ (Active Query).

To learn more about the utility see the following topics:

- [Getting started](#)
- [Working with data model](#)
- [Main window](#)
- [Query panel interface](#)

Getting started

To begin build queries to your database you should describe it first using Data Model Editor. Here are several steps which should be performed:

1. Create new data model.

Select "Data Model | New" menu item. Type name of new data model, select [database gate](#) which will be used and specify the connection to locale database or SQL server.

2. Describe the structure of your model

After finish of the step #1 and pressing OK button you will start your work with Data Model Wizard. Following the instructions placed in wizard you should specify list of the tables included in your data model, link between them and fields which can be used in future queries.

3. Creating a new query

When the wizard will finish its work you see "New query" dialog. Here you should type the name of the query, the name of the file where this query will be stored, description of the query and some additional parameters like "Use DISTINCT in SELECT clause" option.


4. Specifying the conditions

Next step will be to add one or more condition to your query. Just click to "Click here to add new condition" label inside [query panel](#) and select the field, operator (like "equals" or "starts with") and type the value.

5. Specifying the list of result fields

After that you can indicate what columns you would like to see in your query. By default query builder use * in SELECT clause that means that you will get all fields in result. To get only necessary fields - click to "*" label under query panel and add the fields you would like to get using the [Result Fields Editor](#) dialog.

6. Build your query.

Just press "Build Query" () button to get SQL statement and the result record set in "Query Result" panel.

Working with data models

Before starting to build queries to some database (or set of databases) you should describe all necessary parameters of that database (or those databases). The description includes list of tables and views which can be queried, list of fields which take part in the queries, boolean operators used in conditions (such as "is equal to" or "starts with").

The description can contain also some list of predefined values used in conditions and their titles understandable for end-users (e.g. MC - Master Card, CA - California). Such lists can be extracted from the database itself using SQL queries.

So, such database description together with database gate used for connection and list of already built queries constitutes **data model**.

Each data model in KQB is stored in one separate folder: <KQB folder>\dmodels*<data model folder>*

In each moment of time KQB works with one particular data model.

To add new data model you can use *Data Model | New* menu item. To change current data model use *Data Model | Select* menu item.

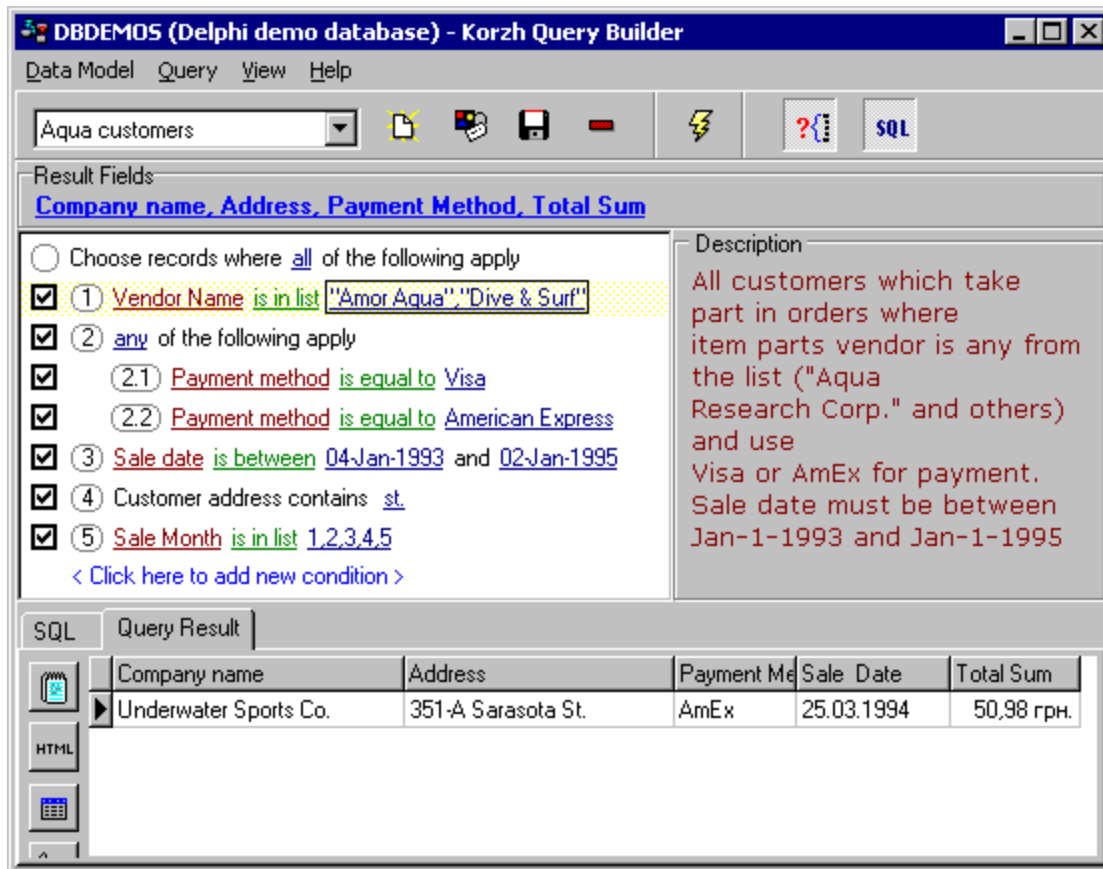
There are two methods to modify some data model. First - through Data Model Editor dialog (menu: *Data Model | Edit*), second - through special Data Model Wizard (menu: *Data Model | Update*).

With the help of Data Model Wizard you can perform only basic operations: add/edit/delete tables, fields and operators and define links between tables. This tool has quite clear user interface, so it is more suitable for non-experienced users.

Data Model Editor provides much more wider abilities for modifications. In addition to all basic operation it allows to define additional virtual fields, boolean operators, custom conditions and a lot of advanced parameters for fields, tables and links.

Main window

You can see the example of Query Builder main window on the picture below.



Main window contains three main parts: menu and toolbar, query panel with the list of result fields at the top and description of the current query at the right side and "Result" panel with two tabs: SQL and "Query result".

KQB toolbar contains "Query" combo box and several buttons which perform necessary operations. Query combobox contains the name of the current query and can be used to select a query from the list of stored queries for current data model.

Other buttons on the toolbar are:



button creates a new query.



button brings up a dialog allowing you to modify query properties such as name, file name and description.



button saves current query.



button deletes current query from data model.



button executes the query showing SQL statement and query result.



button shows or hides query panel.



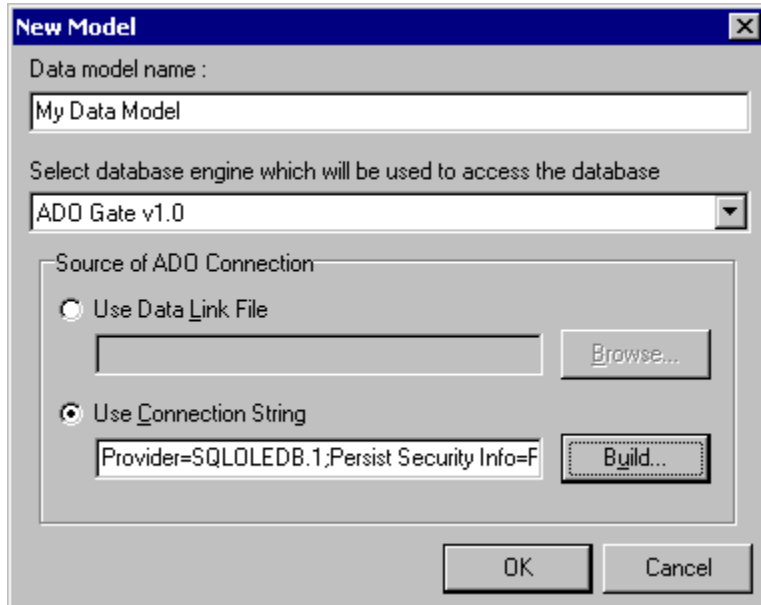
button shows or hides query result panel.

Under the toolbar there is a list of result fields: the fields which will be included into SELECT clause of generated SQL statement. To modify the list - just click on the blue label which represent it and use Result Fields Editor dialog to add, delete or change the result fields.

Create data model dialog

This allows you to type the name of new data model, select database gate which will be used to connect to database(s) and connection parameters (at the bottom of the dialog). The parameters of your connection may be different depending on database gate you use.

As you see at the picture below for ADO gate you should specify connection string but for BDE gate you will need to select BDE alias.



To create new data model just fill all described parameters and press OK button.

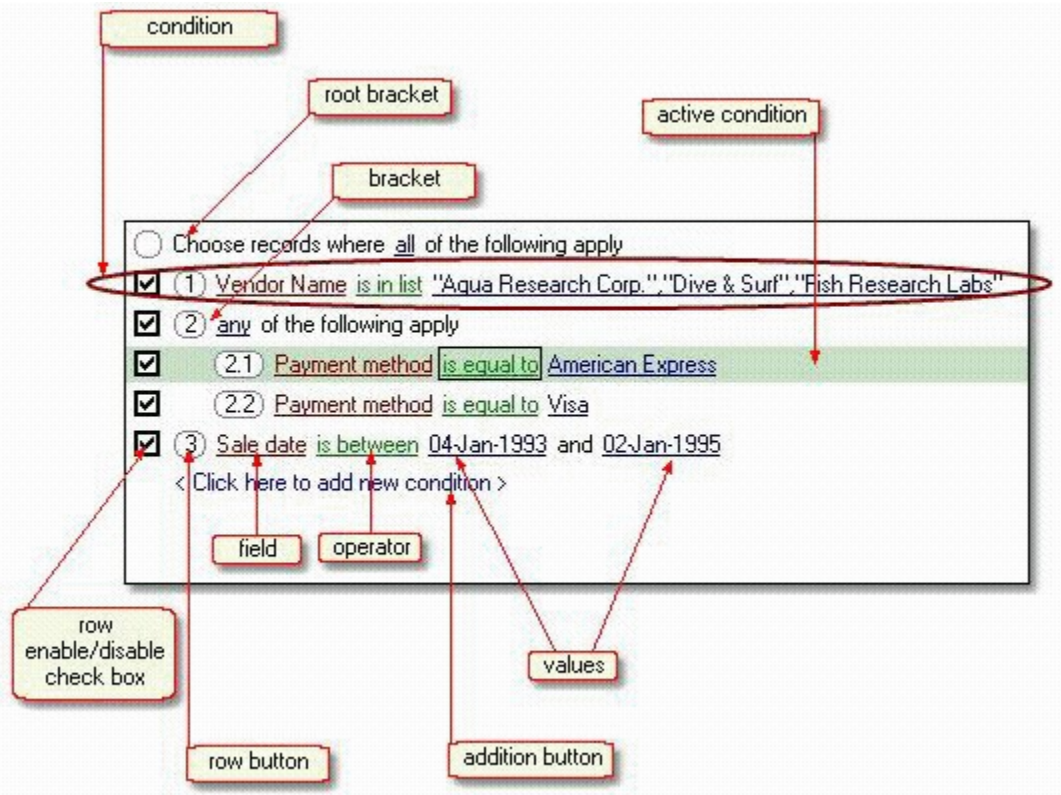
Select data model dialog

Use this dialog to select a data model to open.

All available data models are stored in *dmodels* sub-folder of the main KQB folder. Just select a necessary data model and press OK button.

Query Panel interface

Query panel is the main interface element provided by Korzh Query Builder. It is implemented by **TKQueryPanel** component in Simple Query package or **QueryPanelX** control in Active Query library.



All conditions in query panel are organized in tree with unlimited number of levels. The last nodes in that tree (leaves) represent separate conditions, all other nodes (not last) represent the groups of conditions (brackets). The root of this hierarchy is the "root group" (or "root bracket") which contains all conditions and group of conditions of the first level.

Each bracket has its link type - the way how conditions in the group are linked to each other. There are four possible values of link type: "All", "Any", "None", "Not All".

Each row in query panel represents one condition or title of the conditions group (bracket).

In general, such organization can be simply represented by some phrase in natural language. For example the following query:

```
all
├_ Condition1
├_ Condition2
└_ any
   ├──_ Condition3
   └_ Condition4
```

Can be read as: "we need all data that satisfies Condition1 and Condition2 and one of Condition3 or Condition4".

There two types of conditions: simple conditions and custom conditions. Simple conditions consists of three main parts: a field (one of the fields defined in data model), an operator (such as "is equal to" or "less than"), in and the values. The values list can be empty as in case of "is NULL" operator.

Custom conditions are defined in data model and represented by some textual description and (possibly) one or more values.

Each row contains also "enable/disable" check box and special "row button" which can be used to delete current condition or to add the new one.

The last row in query panel occupy "addition button" which can be used to add one simple condition (left click) or to perform several others task available through context-menu (right-click): add bracket, add custom condition.

Keyboard shortcuts:

Ins Adds a new simple condition after the current condition;

Alt+Ins Brings up context menu. Using this menu you can add new custom condition or bracket;

Ctrl+Del Deletes current row or bracket;

Ctrl+<up arrow> Moves current row up in query panel. In the other words: it changes the places of the current row and the row above it with the same level;

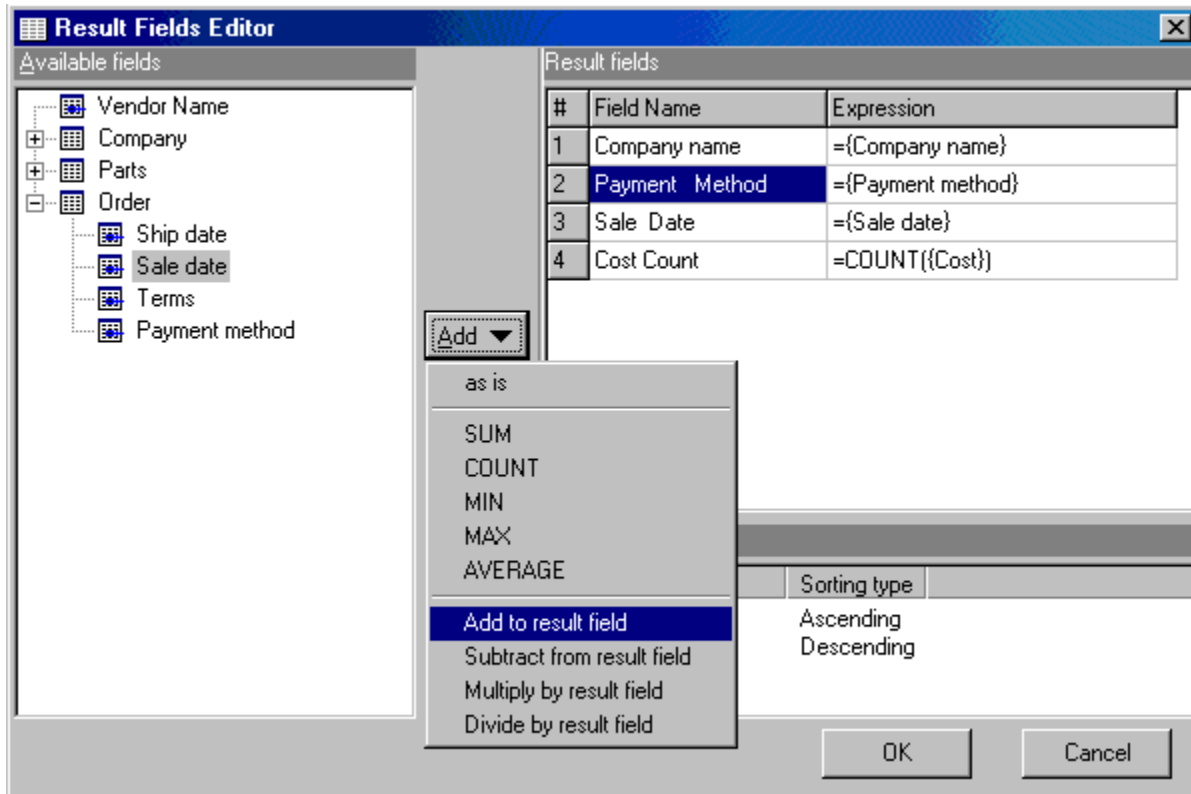
Ctrl+<down arrow> Moves current row down in query panel. In the other words: it changes the places of the current row and the row below it with the same level;

Ctrl+<left arrow> If current condition belongs to some group then it is moved one level up.

Ctrl+<right arrow> If the previous row of the same level is group - then the current condition will be added to that group.

Selecting result fields

To select fields appearing in the query's result click the active area of Result Fields box or choose Query/Edit Result Fields from main menu. Now, let's add another field into query result. Click the field list above query conditions. Result Fields Editor dialog will appear.



Double-click a field you want to add or press "Add" button and select "as is" menu item. You can add more complex result fields using aggregate functions items (SUM, COUNT, etc) and create expressions which contains several data fields combined by arithmetical operations. See [Result Fields Editor dialog](#) for more detailed description of available operations.

Modifying query condition

Below is a sample query condition (Aqua customers query from DBDemos demo data model)

Choose records where **all** of the following apply

- ① VendorName is equal to Aqua Research Corp.
- ② **any** of the following apply
 - ②.1 Payment Method is equal to Visa
 - ②.2 Payment Method is equal to Master Card
- ③ Sale Date is between 01 01 1993 and 01 01 1995

Topmost line shows main criteria of how the subsequent conditions should be treated. Click the active area (now showing all) and choose from a drop-down list. The choices are:

- **all**: only record meeting all of conditions below will get into the result (logical AND)
- **any**: meeting at least one of conditions will grant record's selection (logical OR)
- **none**: record will be rejected if any of conditions is met (logical AND-NOT)
- **not all**: record is included into result set if at least one condition is false; if all of them are met, the record is rejected (logical OR-NOT)

To add a condition, click a circle in the start of the topmost row and choose Add Condition. New condition will appear at the bottom of existing conditions.

To edit a condition (just added or existing) follow the steps:

Click the field's name (black and underlined, e.g., VendorName) and choose appropriate field from the appearing list.

Click the logical operation (here, is equal to) and choose operation you need from the appearing list.

Click the field's value (here, Aqua Research Corp.). If the field is configured to be chosen from list, a list will appear. If the field's value is set to be edited, an edit field will appear so you can type the value from keyboard. For DateTime fields (configured as custom-edited) a Calendar dialog will appear.

For additional information on configuring fields, see [Data Model Editor: fields](#).

To remove a condition click the circle (or oval) showing the condition number and choose Delete Current Row from appearing menu.

Also, you may add a sub condition (called also bracket). In the example above, the sub condition says payment method is Visa or Master Card.

To add a sub condition click the number of any condition on the level you want the sub condition inserted and choose Add Bracket.

Speaking of levels, a sub condition can be inserted into another sub condition. Say, we want to include records having any payment method but having Sale Date and Ship Date later than 1.1.1994. Then we should click the condition numbered 2.2, choose Add Bracket and modify the new sub condition as follows:

Choose records where **all** of the following apply


- ① VendorName is equal to Aqua Research Corp.
- ② **any** of the following apply
 - ②.1 Payment Method is equal to Visa
 - ②.2 Payment Method is equal to Master Card
 - ②.3 **all** of the following apply
 - ②.3.1 Sale Date greater than 01 01 1994
 - ②.3.2 Ship Date greater than 01 01 1994
- ③ Sale Date is between 01 01 1993 and 01 01 1995

Bracket's header (like that of the main bracket) has choices showing what conditions must apply: **all**, **any**, **none** or **not all**.

Query parameters

Each query in the data model has following parameters:

- *Query Name*: short string showing in Select Query listbox
- *File Name*: file to save the query to.
- *Description*: brief description of the query (optional).

The parameters are edited in Query parameters dialog brought up by  button on the main window toolbar or by choosing Query / Edit Params from main menu.

Query results output

After the query is executed, its result can be saved or printed.

From main menu,

- choose Result/Save as text file to save the result as text. Fields are delimited with spaces and included into quotes
- choose Result/Save as HTML to save the result as HTML file. Result is represented as table and fields are formatted according to field definitions (see [Data Model Editor: Fields](#))
- choose Result/Save as DBase table to save the result as DBaseIV DBF file. If needed, field names are changed to meet DBaseIV field naming requirements

Simple Query (Active Query) library

Presented Query Builder application is a demo example of the Simple(Active) Query package.



Simple Query is an end user-oriented SQL builder for Delphi. It is built as a set of 100% VCL components which can be included in your application. Active Query is an ActiveX library which includes two main controls (QueryPanelX and ResultFieldsPanelX) and several auxiliary COM objects.

Simple(Active) Query allows your end users to build their own queries without needing to understand your database design or table relationships. Users describe their queries by selecting plain English (or any other language) terms.

During query design, Simple Query shows long field descriptions instead of actual field names, operators spelled out (e.g. "is equal to") rather than mathematical symbols and field value descriptions (e.g. "MasterCard") rather than actual values ("MC").

What can be queried ?

Simple Query can build queries for any database accessible by the different types of database engines (BDE, ADO, dbExpress, IB Express, ODBC, DBISAM, etc). Active Query library currently works only through ADO.

Simple(Active) Query does not require an end user to know where does each field actually reside. That is, when building a query, the user browses through a field list that can contain fields from multiple tables.

The term "Field" in the Simple Query is not the same as the field in database table.

Each field corresponds with the database table field, but it has also additional parameters like DisplayName (Caption), list of available conditional operators and the editor type. The list of fields is the global list for the database. Each field corresponds to the some entity in the data domain so you don't need to include auxiliary fields like primary keys into the field list.

What you get

Simple(Active) Query generates an SQL SELECT statement designed to capture the data your user specified. You can get this statement and execute it through your DB Engine. Simple(Active) Query automatically builds multiple table queries based on table join information stored in data model. A smart table join method automatically builds a join condition to link the tables whose fields were selected by the user. You can save queries into a text file for further use. Saved query can be loaded later and additionally edited by the end user.

Working with data model

Which fields are shown and how they are presented to your user is determined by special data dictionary customized in [Data Model Editor](#). This dictionary can be stored directly into your program (as resource) or into an INI-like text file.

Data Model Editor allows you to specify field descriptions, captions, editing masks, list of available operations and list for available values. You can maintain multiple data dictionaries and let your user switch between them at run-time. Note that each data dictionary can contain multiple related database tables.

Database

An organized collection of information on a locale storage or on an SQL server.

Database gate

Database gate represents a database engine or method of connection to database data. One database can be connected through different interfaces like ODBC, ADO, BDE, dbExpress or native method for this database.

Each DB gate is a special DLL which encapsulates all operations with database necessary for KQB (like "get list of available databases on server" or "get list of fields for particular table").

All database gates are stored in **dbgates** sub folder of the main KQB folder.

Data Model

Description of a database structure including:

- list of tables
- list of links between tables;
- list of entities of the current data domain named "Fields"
- list of conditional operators used in query conditions;
- list of custom conditions;

