

# **SQL Manager.net**™

EMS® Software Development

## **Advanced Query Builder for RAD Studio VCL User's Manual**

© 1999-2024 EMS Software Development



# **Advanced Query Builder for RAD Studio VCL User's Manual**

© 1999-2024 EMS Software Development

All rights reserved.

This manual documents EMS Advanced Query Builder for RAD Studio VCL, version 3.8.

No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Use of this documentation is subject to the following terms: you may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way.

Document generated on: 19.01.2024

# Table of Contents

|   |           |
|---|-----------|
| <b>Part I Welcome to Advanced Query Builder for RAD Studio VCL!</b> | <b>10</b> |
| Overview .....  | 10        |
| What's new .....  | 11        |
| Installation .....  | 12        |
| Registration .....  | 14        |
| How to register Advanced Query Builder .....                        | 15        |
| Other EMS Products .....  | 16        |
| <b>Part II Advanced Query Builder</b>                               | <b>23</b> |
| <b>TCustomQueryBuilder</b> .....                                    | <b>24</b> |
| TCustomQueryBuilder Reference .....                                 | 24        |
| <b>Properties</b> .....   | <b>25</b> |
| Aggregates .....  | 26        |
| Functions .....   | 27        |
| Keywords .....  | 28        |
| Operators .....   | 29        |
| Options .....   | 30        |
| Quantifiers .....   | 31        |
| Predicates .....  | 32        |
| Palette .....   | 33        |
| Parser .....  | 34        |
| SQL .....   | 35        |
| Style .....   | 36        |
| TabHeight .....   | 37        |
| VisibleTabs .....   | 38        |
| <b>Methods</b> .....  | <b>39</b> |
| DoAddRemoveLink .....   | 40        |
| DoAddRemoveTable .....  | 41        |
| BeginReload .....   | 42        |
| Clear .....   | 43        |
| EndReload .....   | 44        |
| <b>Events</b> .....   | <b>45</b> |
| OnAddRemoveLink .....   | 46        |
| OnAddRemoveTable .....  | 47        |
| OnEnterProcParameter .....  | 48        |
| OnGetTableFields .....  | 49        |
| OnParserError .....   | 50        |
| <b>TFullQueryBuilder</b> .....                                      | <b>51</b> |
| TFullQueryBuilder Reference .....                                   | 51        |
| <b>Properties</b> .....   | <b>52</b> |
| TypeQuery .....   | 53        |
| OnlySelect .....  | 54        |

|   |            |
|---|------------|
| UnionAllByDefault.....                        | 55         |
| QueryBuilders.....                            | 56         |
| <b>Methods</b> .....                          | <b>57</b>  |
| AddQueryBuilder.....                          | 58         |
| DeleteQueryBuilder.....                       | 59         |
| SetActiveQueryBuilder.....                    | 60         |
| <b>Events</b> .....                           | <b>61</b>  |
| OnAddQuery.....                               | 62         |
| <b>TSQLQueryBuilder</b> .....                 | <b>63</b>  |
| <b>TSQLQueryBuilder Reference</b> .....       | <b>63</b>  |
| <b>Properties</b> .....                       | <b>64</b>  |
| Aggregates.....                               | 65         |
| Functions.....                                | 66         |
| Keywords.....                                 | 67         |
| Operators.....                                | 68         |
| Options.....                                  | 69         |
| Palette.....                                  | 70         |
| QuotIdentifiers.....                          | 71         |
| Style.....                                    | 72         |
| TabHeight.....                                | 73         |
| VisibleTabs.....                              | 74         |
| <b>Methods</b> .....                          | <b>75</b>  |
| DoAddRemoveLink.....                          | 76         |
| DoAddRemoveTable.....                         | 77         |
| BeginReload.....                              | 78         |
| Clear.....                                    | 79         |
| EndReload.....                                | 80         |
| <b>Events</b> .....                           | <b>81</b>  |
| OnAddRemoveLink.....                          | 82         |
| OnAddRemoveTable.....                         | 83         |
| OnEnterProcParameter.....                     | 84         |
| OnGetTableFields.....                         | 85         |
| OnParserError.....                            | 86         |
| <b>TInterBaseQueryBuilder</b> .....           | <b>87</b>  |
| <b>TInterBaseQueryBuilder Reference</b> ..... | <b>87</b>  |
| <b>Properties</b> .....                       | <b>88</b>  |
| Aggregates.....                               | 89         |
| Functions.....                                | 90         |
| Keywords.....                                 | 91         |
| Operators.....                                | 92         |
| Options.....                                  | 93         |
| Palette.....                                  | 94         |
| SQLDialect.....                               | 95         |
| SQLServer.....                                | 96         |
| Style.....                                    | 97         |
| TabHeight.....                                | 98         |
| VisibleTabs.....                              | 99         |
| <b>Methods</b> .....                          | <b>100</b> |
| DoAddRemoveLink.....                          | 101        |
| DoAddRemoveTable.....                         | 102        |
| BeginReload.....                              | 103        |
| Clear.....                                    | 104        |
| EndReload.....                                | 105        |

|   |            |
|---|------------|
| <b>Events</b> .....                       | <b>106</b> |
| OnAddRemoveLink.....                      | 107        |
| OnAddRemoveTable.....                     | 108        |
| OnEnterProcParameter .....                | 109        |
| OnGetTableFields.....                     | 110        |
| OnParserError.....                        | 111        |
| <b>TMySQLQueryBuilder</b> .....           | <b>112</b> |
| <b>TMySQLQueryBuilder Reference</b> ..... | <b>112</b> |
| <b>Properties</b> .....                   | <b>113</b> |
| Aggregates.....                           | 114        |
| Functions.....                            | 115        |
| Keywords.....                             | 116        |
| Operators.....                            | 117        |
| Options.....                              | 118        |
| Palette.....                              | 119        |
| QuoteIdentifiers.....                     | 120        |
| Style.....                                | 121        |
| TabHeight.....                            | 122        |
| VisibleTabs .....                         | 123        |
| <b>Methods</b> .....                      | <b>124</b> |
| DoAddRemoveLink.....                      | 125        |
| DoAddRemoveTable.....                     | 126        |
| BeginReload.....                          | 127        |
| Clear.....                                | 128        |
| EndReload.....                            | 129        |
| <b>Events</b> .....                       | <b>130</b> |
| OnAddRemoveLink.....                      | 131        |
| OnAddRemoveTable.....                     | 132        |
| OnEnterProcParameter .....                | 133        |
| OnGetTableFields.....                     | 134        |
| OnParserError.....                        | 135        |
| <b>TPgSQLQueryBuilder</b> .....           | <b>136</b> |
| <b>TPgSQLQueryBuilder Reference</b> ..... | <b>136</b> |
| <b>Properties</b> .....                   | <b>137</b> |
| Aggregates.....                           | 138        |
| Functions.....                            | 139        |
| Keywords.....                             | 140        |
| Operators.....                            | 141        |
| Options.....                              | 142        |
| Palette.....                              | 143        |
| Style.....                                | 144        |
| TabHeight.....                            | 145        |
| UseNamespaces.....                        | 146        |
| VisibleTabs .....                         | 147        |
| <b>Methods</b> .....                      | <b>148</b> |
| DoAddRemoveLink.....                      | 149        |
| DoAddRemoveTable.....                     | 150        |
| BeginReload.....                          | 151        |
| Clear.....                                | 152        |
| EndReload.....                            | 153        |
| <b>Events</b> .....                       | <b>154</b> |
| OnAddRemoveLink.....                      | 155        |
| OnAddRemoveTable.....                     | 156        |

|   |            |
|---|------------|
| OnEnterProcParameter.....                 | 157        |
| OnGetTableFields.....                     | 158        |
| OnParserError.....                        | 159        |
| <b>TMSSQLQueryBuilder .....</b>           | <b>160</b> |
| <b>TMSSQLQueryBuilder Reference .....</b> | <b>160</b> |
| <b>Properties .....</b>                   | <b>161</b> |
| Aggregates.....                           | 162        |
| AlterQuotes.....                          | 163        |
| Functions.....                            | 164        |
| Keyw ords.....                            | 165        |
| Operators.....                            | 166        |
| Options.....                              | 167        |
| Palette.....                              | 168        |
| Style.....                                | 169        |
| TabHeight.....                            | 170        |
| VisibleTabs .....                         | 171        |
| <b>Methods .....</b>                      | <b>172</b> |
| DoAddRemoveLink.....                      | 173        |
| DoAddRemoveTable.....                     | 174        |
| BeginReload.....                          | 175        |
| Clear.....                                | 176        |
| EndReload.....                            | 177        |
| <b>Events .....</b>                       | <b>178</b> |
| OnAddRemoveLink.....                      | 179        |
| OnAddRemoveTable.....                     | 180        |
| OnEnterProcParameter .....                | 181        |
| OnGetTableFields.....                     | 182        |
| OnParserError.....                        | 183        |
| <b>TDb2QueryBuilder .....</b>             | <b>184</b> |
| <b>TDb2QueryBuilder Reference .....</b>   | <b>184</b> |
| <b>Properties .....</b>                   | <b>185</b> |
| Aggregates.....                           | 186        |
| Functions.....                            | 187        |
| Keyw ords.....                            | 188        |
| Operators.....                            | 189        |
| Options.....                              | 190        |
| Palette.....                              | 191        |
| Style.....                                | 192        |
| TabHeight.....                            | 193        |
| UseNamespaces.....                        | 194        |
| VisibleTabs .....                         | 195        |
| <b>Methods .....</b>                      | <b>196</b> |
| DoAddRemoveLink.....                      | 197        |
| DoAddRemoveTable.....                     | 198        |
| BeginReload.....                          | 199        |
| Clear.....                                | 200        |
| EndReload.....                            | 201        |
| <b>Events .....</b>                       | <b>202</b> |
| OnAddRemoveLink.....                      | 203        |
| OnAddRemoveTable.....                     | 204        |
| OnEnterProcParameter .....                | 205        |
| OnGetTableFields.....                     | 206        |
| OnParserError.....                        | 207        |

|   |            |
|---|------------|
| <b>TDbiQueryBuilder</b> .....           | <b>208</b> |
| <b>TDbiQueryBuilder Reference</b> ..... | <b>208</b> |
| <b>Properties</b> .....                 | <b>209</b> |
| Aggregates.....                         | 210        |
| Functions.....                          | 211        |
| Keywords.....                           | 212        |
| Operators.....                          | 213        |
| Options.....                            | 214        |
| Palette.....                            | 215        |
| Style.....                              | 216        |
| TabHeight.....                          | 217        |
| VisibleTabs.....                        | 218        |
| <b>Methods</b> .....                    | <b>219</b> |
| DoAddRemoveLink.....                    | 220        |
| DoAddRemoveTable.....                   | 221        |
| BeginReload.....                        | 222        |
| Clear.....                              | 223        |
| EndReload.....                          | 224        |
| <b>Events</b> .....                     | <b>225</b> |
| OnAddRemoveLink.....                    | 226        |
| OnAddRemoveTable.....                   | 227        |
| OnEnterProcParameter.....               | 228        |
| OnGetTableFields.....                   | 229        |
| OnParserError.....                      | 230        |

## Part III Units

## 232

|                                   |            |
|-----------------------------------|------------|
| <b>QBWindow unit</b> .....        | <b>232</b> |
| <b>TQBPalette class</b> .....     | <b>233</b> |
| Properties.....                   | 234        |
| ActiveCondition.....              | 235        |
| ProcClient .....                  | 236        |
| TableClient .....                 | 237        |
| Field .....                       | 238        |
| Group .....                       | 239        |
| Operation .....                   | 240        |
| Predicate .....                   | 241        |
| SubQuery .....                    | 242        |
| <b>TQBStyle class</b> .....       | <b>243</b> |
| Properties.....                   | 244        |
| OldStyleLook .....                | 245        |
| ButtonStyle .....                 | 246        |
| ObjectStyle .....                 | 247        |
| TableStyle .....                  | 248        |
| <b>TQBObjectStyle class</b> ..... | <b>249</b> |
| Properties.....                   | 250        |
| BorderKind .....                  | 251        |
| Flat .....                        | 252        |
| FlatButtons .....                 | 253        |
| <b>TQueryBuilder</b> .....        | <b>254</b> |
| Properties.....                   | 255        |
| Data .....                        | 256        |
| <b>TQueryBuilders</b> .....       | <b>257</b> |
| Properties.....                   | 258        |

|                                |            |
|--------------------------------|------------|
| Items .....                    | 259        |
| Methods .....                  | 260        |
| Add .....                      | 261        |
| FindByName .....               | 262        |
| <b>QBFloatTable unit .....</b> | <b>263</b> |
| <b>TQBTable class .....</b>    | <b>264</b> |
| Properties.....                | 265        |
| Alias .....                    | 266        |
| Caption .....                  | 267        |
| Expand .....                   | 268        |
| Height .....                   | 269        |
| ItemHeight .....               | 270        |
| Left .....                     | 271        |
| TableName .....                | 272        |
| Top .....                      | 273        |
| Width .....                    | 274        |
| <b>QBIBWindow unit .....</b>   | <b>275</b> |
| <b>QBMyWindow unit .....</b>   | <b>276</b> |
| <b>QBPgWindow unit .....</b>   | <b>277</b> |
| <b>QBMSWindow unit .....</b>   | <b>278</b> |
| <b>QBDb2Window unit .....</b>  | <b>279</b> |
| <b>QBDbiWindow unit .....</b>  | <b>280</b> |



**Part**



# 1 Welcome to Advanced Query Builder for RAD Studio VCL!

## 1.1 Overview

**EMS Advanced Query Builder for RAD Studio VCL** is a component for Delphi and C++ Builder intended for visual building SQL statements for the SELECT, INSERT, UPDATE and DELETE clauses. It allows you to build new queries visually and/or graphically represent the existing queries in your own applications. The suite includes components for working with standard SQL and a number of dialects used in most popular RDBMS.

Advanced Query Builder enables users to make up large and complicated SQL queries with unions and subqueries for different servers even without any knowledge of the SQL syntax.

### Key features

- Support for 64-bit Windows target platform
- Standard SQL, InterBase/Firebird, MySQL, PostgreSQL, SQL Server, Oracle, and DB2 support
- Detailed help system and a demo application for quicker mastering the product
- Powerful component and property editors which admit to setting many query building parameters at the design-time easily
- High productivity even on slow computers
- Powerful SQL parser which allows you to represent complicate SQL statements visually
- Creating TxxxQueryBuilder components for working with the specified server easily
- Full customization of font colors, object appearance, etc.
- Delphi 2010, XE-XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11 Alexandria, 12 Athens, and C++ Builder 2010, XE-XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11 Alexandria, 12 Athens support

### Product information

|                       |   |
|-----------------------|---|
| Homepage              | <a href="https://www.sqlmanager.net/products/tools/querybuilder">https://www.sqlmanager.net/products/tools/querybuilder</a>         |
| Support Ticket System | <a href="https://www.sqlmanager.net/support">https://www.sqlmanager.net/support</a>   |
| Register on-line at   | <a href="https://www.sqlmanager.net/products/tools/querybuilder/buy">https://www.sqlmanager.net/products/tools/querybuilder/buy</a> |

## 1.2 What's new

**Version**

Advanced Query Builder for RAD Studio VCL 3.14

**Release date**

January 19, 2024

**What's new in Advanced Query Builder for RAD Studio VCL 3.14?**

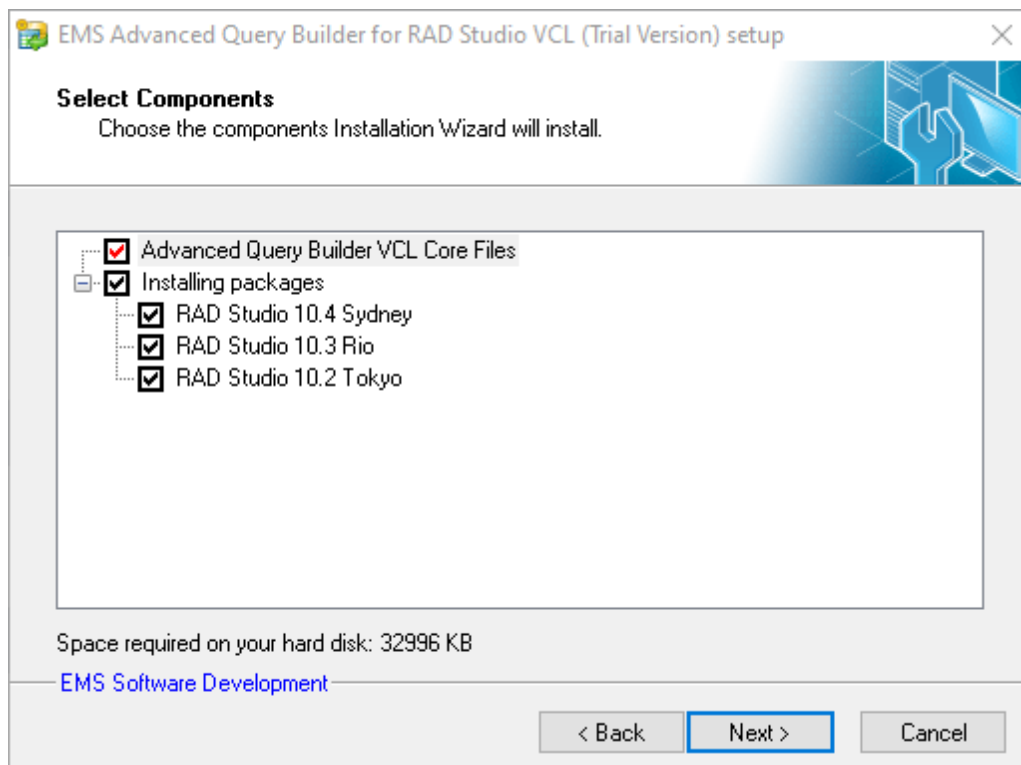
- Support for RAD Studio 12 Athens implemented.
- Fixed parsing of parameters in the expression with IN operator.
- Fixed removal of table alias in CASE expression.
- TcxScrollBar in DevExpress version of components.
- Fixed duplicated columns in SELECT query.
- Fixed building components with {\$T+} compiler option enabled.
- Fixed parsing of CASE operator.
- Fixed hang on parsing GROUP BY clause.
- Parentheses in JOIN clauses are optional for MS SQL Server/MS Access now.
- Fixed parsing of LEFT JOIN clause for MS SQL Server.
- Fixed paths for 32-bit Clang compiler in RAD Studio options.
- Other minor fixes.

## 1.3 Installation

To install the **trial version** of **Advanced Query Builder for RAD Studio VCL** onto your system:

- download the distribution package of **Advanced Query Builder for RAD Studio VCL** from the [download page](#) available at our website;
- unzip the downloaded file to any local directory, e.g. *C:\unzipped*;
- close all currently opened Delphi and/or C++ Builder IDEs, if any;
- run the executable setup file from the local directory and follow the instructions of the installation wizard.

During the installation you will need to select the packages to install.



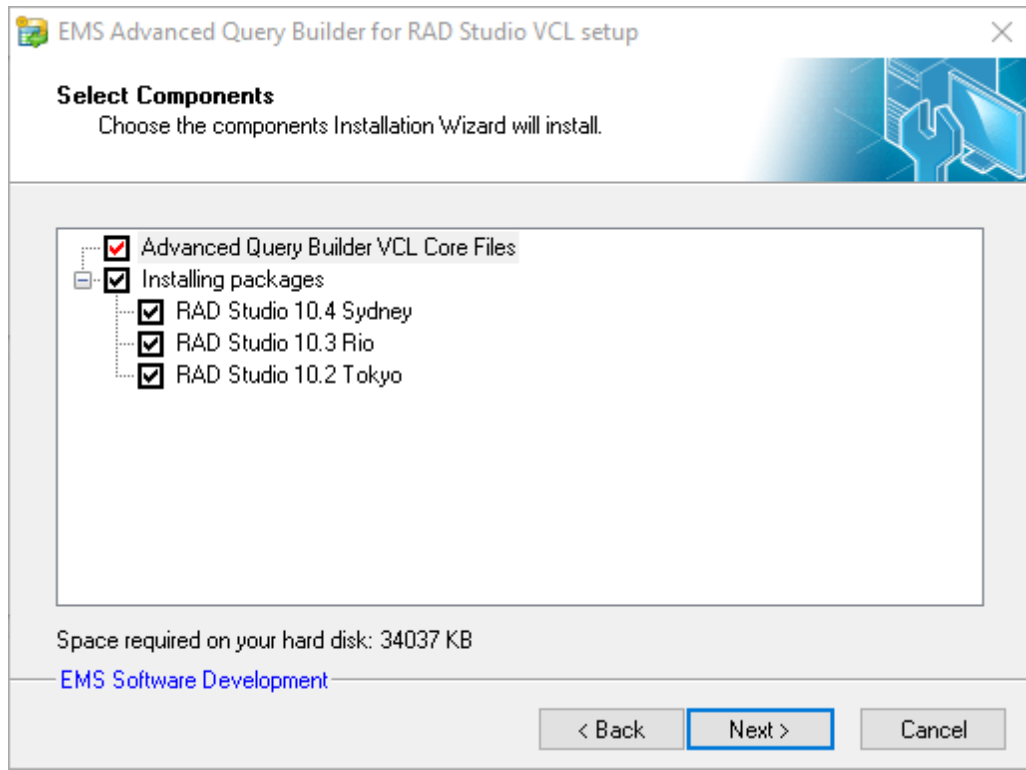
When you are done, you can finish installation of the **trial version** of **Advanced Query Builder for RAD Studio VCL**.

To install the **full version** of **Advanced Query Builder for RAD Studio VCL** onto your system:

- download the distribution package of **Advanced Query Builder for RAD Studio VCL** from the [download page](#) available at our website;
- unzip the downloaded file to any local directory, e.g. *C:\unzipped*;
- close all currently opened Delphi and/or C++ Builder IDEs, if any;
- run the executable setup file from the local directory and follow the instructions of the installation wizard.

Enter valid registration information in the appropriate boxes: **Registration name** and **Registration Key**. See [details](#) on getting this information.

During the installation you will need to select the packages to install.



When you are done, you can finish installation of the **full version** of **Advanced Query Builder for RAD Studio VCL**.

**Note:** If the above given instructions have been insufficient for successful installation of the component suite, please refer to the *readme.1st* file distributed with the product.

## 1.4 Registration

All purchases are provided by **Digital River** registration service. The **Digital River** order process is protected via a secure connection and makes on-line ordering by credit/debit card quick and safe.

**Digital River** is a global e-commerce provider for software and shareware sales via the Internet. It accepts payments in US Dollars, Euros, Pounds Sterling, Japanese Yen, Australian Dollars, Canadian Dollars or Swiss Franks by Credit Card (Visa, MasterCard/ EuroCard, American Express, Diners Club), Bank/Wire Transfer, Check or Cash.

If you want to review your order information, or you have questions about ordering or payments please visit our [Customer Care Center](#), provided by **Digital River**.

Please note that all of our products are delivered via ESD (Electronic Software Delivery) only. After purchase you will be able to immediately download the registration keys or passwords. Also you will receive a copy of registration keys or passwords by email. Please make sure to enter a valid email address in your order. If you have not received the keys within 2 hours, please, contact us at [sales@sqlmanager.net](mailto:sales@sqlmanager.net).

| Product distribution  | MyCommerce/Digital River      |
|---|-------------------------------|
| <b>Advanced Query Builder for RAD Studio VCL</b> Component Full version (with sources)* | <a href="#">Register Now!</a> |
| <b>Advanced Query Builder for RAD Studio VCL</b> Component Trial version                | <a href="#">Download Now!</a> |

\* **EMS Maintenance Program** provides the following benefits:

- Free software bug fixes, enhancements, updates and upgrades during the maintenance period
- Free unlimited communications with technical staff for the purpose of reporting Software failures
- Free reasonable number of communications for the purpose of consultation on operational aspects of the software

After your maintenance expires you will not be able to update your software or get technical support. To protect your investments and have your software up-to-date, you need to renew your maintenance.

You can easily reinitiate/renew your maintenance with our on-line, speed-through Maintenance Reinstatement/Renewal Interface. After reinitiating/renewal you will receive a confirmation e-mail with all the necessary information.

## 1.5 How to register Advanced Query Builder

To register your newly purchased copy of **EMS Advanced Query Builder for RAD Studio VCL**, perform the following steps:

- receive the notification letter from **Digital River** with the registration info;
- enter the **Registration Name** and the **Registration Key** from this letter while [installing](#) the **full version** of the product.

---

**See also:**

[Registration](#)

## 1.6 Other EMS Products

### Quick navigation



[MySQL](#)



[Microsoft SQL Server](#)



[PostgreSQL](#)



[InterBase / FireBird](#)



[Oracle](#)



[IBM DB2](#)



[Tools & components](#)

### MySQL



#### [SQL Management Studio for MySQL](#)

EMS SQL Management Studio for MySQL is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



#### [SQL Manager for MySQL](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



#### [Data Export for MySQL](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more.



#### [Data Import for MySQL](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



#### [Data Pump for MySQL](#)

Migrate from most popular databases (MySQL, PostgreSQL, Oracle, DB2, InterBase/Firebird, etc.) to MySQL.



#### [Data Generator for MySQL](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



#### [DB Comparer for MySQL](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



#### [DB Extract for MySQL](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



#### [SQL Query for MySQL](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



#### [Data Comparer for MySQL](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)



## Microsoft SQL Server



### [SQL Management Studio for SQL Server](#)

EMS SQL Management Studio for SQL Server is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



### [EMS SQL Backup for SQL Server](#)

Perform backup and restore, log shipping and many other regular maintenance tasks on the whole set of SQL Servers in your company.



### [SQL Administrator for SQL Server](#)

Perform administrative tasks in the fastest, easiest and most efficient way. Manage maintenance tasks, monitor their performance schedule, frequency and the last execution result.



### [SQL Manager for SQL Server](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



### [Data Export for SQL Server](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more



### [Data Import for SQL Server](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



### [Data Pump for SQL Server](#)

Migrate from most popular databases (MySQL, PostgreSQL, Oracle, DB2, InterBase/Firebird, etc.) to Microsoft® SQL Server™.



### [Data Generator for SQL Server](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



### [DB Comparer for SQL Server](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



### [DB Extract for SQL Server](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



### [SQL Query for SQL Server](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



### [Data Comparer for SQL Server](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)

## PostgreSQL



### [SQL Management Studio for PostgreSQL](#)

EMS SQL Management Studio for PostgreSQL is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



### [EMS SQL Backup for PostgreSQL](#)

Creates backups for multiple PostgreSQL servers from a single console. You can use automatic backup tasks with advanced schedules and store them in local or remote folders or cloud storages



### [SQL Manager for PostgreSQL](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



### [Data Export for PostgreSQL](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more



### [Data Import for PostgreSQL](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



### [Data Pump for PostgreSQL](#)

Migrate from most popular databases (MySQL, SQL Server, Oracle, DB2, InterBase/Firebird, etc.) to PostgreSQL.



### [Data Generator for PostgreSQL](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



### [DB Comparer for PostgreSQL](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



### [DB Extract for PostgreSQL](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



### [SQL Query for PostgreSQL](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



### [Data Comparer for PostgreSQL](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)

## **InterBase / Firebird**



### [SQL Management Studio for InterBase/Firebird](#)

EMS SQL Management Studio for InterBase and Firebird is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



### [SQL Manager for InterBase/Firebird](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



### [Data Export for InterBase/Firebird](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more



### [Data Import for InterBase/Firebird](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



### [Data Pump for InterBase/Firebird](#)

Migrate from most popular databases (MySQL, SQL Server, Oracle, DB2, PostgreSQL, etc.) to InterBase/Firebird.



### [Data Generator for InterBase/Firebird](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



### [DB Comparer for InterBase/Firebird](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



### [DB Extract for InterBase/Firebird](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



### [SQL Query for InterBase/Firebird](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



### [Data Comparer for InterBase/Firebird](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)

## Oracle



### [SQL Management Studio for Oracle](#)

EMS SQL Management Studio for Oracle is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!



### [SQL Manager for Oracle](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



### [Data Export for Oracle](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more.



### [Data Import for Oracle](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via

user-friendly wizard interface.



#### [Data Pump for Oracle](#)

Migrate from most popular databases (MySQL, PostgreSQL, MySQL, DB2, InterBase/Firebird, etc.) to Oracle



#### [Data Generator for Oracle](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



#### [DB Comparer for Oracle](#)

Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.



#### [DB Extract for Oracle](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



#### [SQL Query for Oracle](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.



#### [Data Comparer for Oracle](#)

Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

[Scroll to top](#)

## IBM DB2



#### [SQL Manager for DB2](#)

Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.



#### [Data Export for DB2](#)

Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more.



#### [Data Import for DB2](#)

Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.



#### [Data Pump for DB2](#)

Migrate from most popular databases (MySQL, PostgreSQL, Oracle, MySQL, InterBase/Firebird, etc.) to DB2



#### [Data Generator for DB2](#)

Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.



#### [DB Extract for DB2](#)

Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.



#### [SQL Query for DB2](#)

Analyze and retrieve your data, build your queries visually, work with query plans, build charts

based on retrieved data quickly and more.

[Scroll to top](#)

## Tools & components



### [Advanced Data Export for RAD Studio VCL](#)

Advanced Data Export for RAD Studio VCL allows you to save your data in the most popular office programs formats.



### [Advanced Data Export .NET](#)

Advanced Data Export .NET is a component for Microsoft Visual Studio .NET that will allow you to save your data in the most popular data formats for the future viewing, modification, printing or web publication. You can export data into MS Access, MS Excel, MS Word (RTF), PDF, TXT, DBF, CSV and more! There will be no need to waste your time on tiresome data conversion - Advanced Data Export will do the task quickly and will give the result in the desired format.



### [Advanced Data Import for RAD Studio VCL](#)

Advanced Data Import for RAD Studio VCL will allow you to import your data to the database from files in the most popular data formats.



### [Advanced PDF Generator for RAD Studio](#)

Advanced PDF Generator for RAD Studio gives you an opportunity to create PDF documents with your applications written on Delphi or C++ Builder.



### [Advanced Query Builder for RAD Studio VCL](#)

Advanced Query Builder for RAD Studio VCL is a powerful component for Delphi and C++ Builder intended for visual building SQL statements for the SELECT, INSERT, UPDATE and DELETE clauses.



### [Advanced Excel Report for RAD Studio](#)

Advanced Excel Report for RAD Studio is a powerful band-oriented generator of template-based reports in MS Excel.



### [Advanced Localizer for RAD Studio VCL](#)

Advanced Localizer for RAD Studio VCL is an indispensable component for Delphi for adding multilingual support to your applications.

[Scroll to top](#)

**Part**



## 2 Advanced Query Builder

**EMS Advanced Query Builder for RAD Studio VCL** represents a set of tools for efficient query building.

From a programmer's point of view the suite represents a homomorphic hierarchy of classes with the common ancestor [TCustomQueryBuilder](#). It contains properties and methods which are common for all the descendant classes. Beside the basic properties, methods and events, some specific characteristics are included in descendant classes.

**Advanced Query Builder for RAD Studio VCL** provides a collection of the following components:

| <b>Component</b>                       | <b>Brief description</b>   |
|--|--|
| <a href="#">TCustomQueryBuilder</a>    | Common component for query building  |
| <a href="#">TFullQueryBuilder</a>      | Represents complex SQL query. It holds collection of TCustomQueryBuilder components representing single SELECT queries |
| <a href="#">TSQLQueryBuilder</a>       | Provides building queries with standard SQL syntax   |
| <a href="#">TInterBaseQueryBuilder</a> | Provides building queries for InterBase/FireBird   |
| <a href="#">TMySQLQueryBuilder</a>     | Provides building queries for MySQL  |
| <a href="#">TPgSQLQueryBuilder</a>     | Provides building queries for PostgreSQL   |
| <a href="#">TMSSQLQueryBuilder</a>     | Provides building queries for Microsoft® SQL Server™   |
| <a href="#">TDB2QueryBuilder</a>       | Provides building queries for DB2  |
| <a href="#">TDBIQueryBuilder</a>       | Provides building queries for DBISAM   |

## 2.1 TCustomQueryBuilder

### 2.1.1 TCustomQueryBuilder Reference

#### Unit

[QBWindow](#)


#### Description



The *TCustomQueryBuilder* component is the basic component of the collection. It contains properties, methods and events that are common for all its descendants.



## 2.1.2 Properties

▶ Run-time only

 Key properties

- ▶  [Aggregates](#)
- ▶  [Functions](#)
- ▶  Images
- ▶  [Keywords](#)
- ▶  [Operators](#)
- ▶  [Options](#)
- ▶  [Predicates](#)
- ▶  [Quantifiers](#)
- ▶  [Palette](#)
- ▶  [Parser](#)
- ▶  [SQL](#)
- ▶  [Style](#)
- ▶  [TabHeight](#)
- ▶  [VisibleTabs](#)

### 2.1.2.1 Aggregates

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Aggregates: TStrings;
```

**Description**

The *Aggregates* property contains the list of all the Aggregate functions, available in the 'Grouping criterions' area for use in the query.

---

**See also:**

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.1.2.2 Functions

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Functions: TStrings;
```

**Description**

The *Functions* property contains the list of the SQL functions, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.1.2.3 Keywords

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Keywords: TStrings;
```

**Description**

The *Keywords* property contains the list of all the SQL keywords, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

#### 2.1.2.4 Operators

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Operators: TQBOperators;
```

**Description**

The *Operators* property contains all the operators, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.1.2.5 Options

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBOptions = set of (qboRowSelect, qboDragFieldName, qboHideSelection, qboShowFieldTypes)
```

```
property Options: TQBOptions;
```

#### Description

The *Options* property is a set of suboptions which define the following options of the component:

|                                 |  |
|---------------------------------|--|
| <i>qboRowSelect</i>             | selects the whole row in the 'Criteria' area   |
| <i>qboDragFieldName</i>         | shows the field name on dragging the field   |
| <i>qboHideSelection</i>         | hides the selected items if the object is inactive   |
| <i>qboShowFieldTypes</i>        | shows the field types by hint  |
| <i>qboClearTabsOnTypeChange</i> | indicates to clear 'Criterion' tabs when changing type of the query (SELECT, INSERT, etc.) |

---

#### See also:

[Palette property](#)

### 2.1.2.6 Quantifiers

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Quantifiers: TStrings;
```

**Description**

The *Quantifiers* property contains the list of the SQL quantifiers, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

### 2.1.2.7 Predicates

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Predicates: TStrings;
```

**Description**

The *Predicates* property contains the list of the SQL predicates, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Quantifiers property](#)



### 2.1.2.8 Palette

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

`property` Palette: [TQBPalette](#);

**Description**

The *Palette* property defines various colors, used by the component.

---

**See also:**

[Options property](#)

### 2.1.2.9 Parser

**Applies to**  
[TCustomQueryBuilder](#) component

**Declaration**  
`property Parser: TQBSelectParser;`

#### **Description**

The *Parser* property is used for parsing the SELECT statement, assigned to the [SQL](#) property, for defining different objects (tables, fields, conditions, etc.) used in the query. This property is read-only.

---

**See also:**  
[SQL property](#)

#### 2.1.2.10 SQL

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property SQL: string;
```

**Description**

The *SQL* property contains the result SQL statement to be executed on the server.

---

**See also:**

[Parser property](#)

### 2.1.2.11 Style

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Style: TQBStyle;
```

**Description**

The *Style* property defines the appearance of the 'Criteria' area buttons and 'Builder' area objects.

---

**See also:**

[TQBStyle class](#)

### 2.1.2.12 TabHeight

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property TabHeight: Integer;
```

**Description**

The *TabHeight* property defines the height of the condition panels.

---

**See also:**

[VisibleTabs property](#)

### 2.1.2.13 VisibleTabs

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBVisibleTab = (qbtCriteriaions, qbtSelection, qbtGroupCriteriaions, qbtSorting, qbtData);  
TQBVisibleTabs = set of TQBVisibleTab;
```

```
property VisibleTabs: TQBVisibleTabs;
```

**Description**


The *VisibleTabs* property defines which of the Advanced QueryBuilder tabs ('Criteriaions', 'Selection', Grouping Criteriaions', 'Sorting', 'Data') are visible.






---

**See also:**

[TabHeight property](#)

## 2.1.3 Methods

 Key methods

-  [DoAddRemoveLink](#)
-  [DoAddRemoveTable](#)
-  [BeginReload](#)
-  [Clear](#)
-  [EndReload](#)

### 2.1.3.1 DoAddRemoveLink

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveLink(Link: TQBLink; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveLink* method to add or remove link to the work area. The Link parameter is the TQBLink object, which defines the properties of the link to add or remove. If Action is qbaAdd, then the link, specified by the Link parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveTable method](#)



### 2.1.3.2 DoAddRemoveTable

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveTable(Table: TQBTable; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveTable* method to add or remove table to the work area. The Table parameter is the TQBTable object, which defines the properties of the table to add or remove. If Action is qbaAdd, then the table, specified by the Table parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveLink method](#)

### 2.1.3.3 BeginReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure BeginReload; virtual;
```

**Description**

The *BeginReload* method prepares the 'Criteria' and 'Grouping Criteria' areas for making any changes. You must call this method each time you want to change the [Operators](#) property and call [EndReload](#) when all the changes are done.

---

**See also:**

[Operators property](#)

[EndReload method](#)

#### 2.1.3.4 Clear

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure Clear;
```

**Description**

The *Clear* method clears the work area and all the condition panels.

---

**See also:**

[DoAddRemoveLink method](#)

[DoAddRemoveTable method](#)

### 2.1.3.5 EndReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure EndReload; virtual;
```

**Description**

The *EndReload* method should be called when the changes in the 'Criteria' panel, started by the [BeginReload](#) method are done.

---






**See also:**

[Operators property](#)

[BeginReload method](#)

## 2.1.4 Events

### Key events

-  [OnAddRemoveLink](#)
-  [OnAddRemoveTable](#)
-  [OnEnterProcParameter](#)
-  [OnGetTableFields](#)
-  [OnParserError](#)

#### 2.1.4.1 OnAddRemoveLink

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveLinkEvent = procedure(Sender: TObject; Link: TQBLink; Action: TQBAction; var
```

```
property OnAddRemoveLink: TAddRemoveLinkEvent;
```

##### Description

The *OnAddRemoveLink* event takes place when a link is added/removed to/from the component work area. The Link parameter defines the properties of the link to add/remove. Action defines if the link is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveTable event](#)

#### 2.1.4.2 OnAddRemoveTable

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveTableEvent = procedure(Sender: TObject; Table: TQBTable; Action: TQBAction
```

```
property OnAddRemoveTable: TAddRemoveTableEvent;
```

##### Description

The *OnAddRemoveTable* event takes place when a table is added/removed to/from the component work area. The Table parameter defines the properties of the table to add/remove. Action defines if the table is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveLink event](#)

### 2.1.4.3 OnEnterProcParameter

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TEnterProcParameter = procedure(Sender: TObject; SProcedure: TQBTable; ListItem: TQ
```

```
property OnEnterProcParameter: TEnterProcParameter;
```

**Description**

The *OnEnterProcParameter* event takes place before the procedure input parameter is added.

---

**See also:**

[OnGetTableFields event](#)



#### 2.1.4.4 OnGetTableFields

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TGetTableFieldsEvent = procedure(const NameOfTable: string; var ADataSet: TDataSet;
```

```
property OnGetTableFields: TGetTableFieldsEvent;
```

**Description**

The *OnGetTableFields* event takes place, when you set the [SQL](#) property in run-time. Constant parameter *NameOfTable* is the table name, used in the SELECT ... FROM statement. Using parameter *ADataSet* you can define the dataset, containing data to select from, and using *AParams* you can define the dataset parameters.

---

**See also:**

[OnEnterProcParameter](#)

#### 2.1.4.5 OnParserError

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property OnParserError: TQBParserErrorEvent;
```

**Description**

The *OnParserError* event takes place when parsing the [SQL property](#) causes an error in the TQBParser object. When error occurs, you receive the following information: error code, line number, line position and token.

---

**See also:**

[Parser property](#)

[SQL property](#)

## 2.2 TFullQueryBuilder

### 2.2.1 TFullQueryBuilder Reference

**Unit**

[QBWindow](#)

**Description**

The *TFullQueryBuilder* component representing complex SQL query. It holds collection of [TCustomQueryBuilder](#) components representing single SELECT queries.

## 2.2.2 Properties

▶ Run-time only

🔑 Key properties

- ▶ 🔑 [TypeQuery](#)
- ▶ 🔑 [OnlySelect](#)
- ▶ 🔑 [UnionAllByDefault](#)
- ▶ 🔑 [QueryBuilders](#)
- ▶ [Aggregates](#)
- ▶ [Functions](#)
- ▶ [Keywords](#)
- ▶ [Operators](#)
- ▶ [Predicates](#)
- ▶ [Quantifiers](#)
- ▶ [Options](#)
- ▶ [Palette](#)
- ▶ [QuoteIdentifiers](#)
- ▶ [Style](#)
- ▶ [TabHeight](#)
- ▶ [VisibleTabs](#)

### 2.2.2.1 TypeQuery

**Applies to**

[TFullQueryBuilder](#) component

**Declaration**

**type**

```
TQBTypeQuery = (tqSelect, tqInsert, tqUpdate, tqDelete);
```

**property** TypeQuery: TQBTypeQuery;

**Description**

The *TypeQuery* property determines SQL statement represented by the component. There are SELECT, INSERT, UPDATE and DELETE statements available for building.

---

**See also:**

[OnlySelect property](#)

[UnionAllByDefault property](#)

### 2.2.2.2 OnlySelect

**Applies to**

[TFullQueryBuilder](#) component

**Declaration**

```
property OnlySelect: Boolean;
```

**Description**

The *OnlySelect* property restricts building INSERT, UPDATE and DELETE SQL statements being set to True.

---

**See also:**

[TypeQuery property](#)

[UnionAllByDefault property](#)

### 2.2.2.3 UnionAllByDefault

**Applies to**

[TFullQueryBuilder](#) component

**Declaration**

```
property UnionAllByDefault: Boolean;
```

**Description**

The *UnionAllByDefault* property specifies type of combining results of two joined SELECT statements. While set to True the default type is UNION ALL, otherwise - UNION.

---

**See also:**

[OnlySelect property](#)

[TypeQuery property](#)

#### 2.2.2.4 QueryBuilders

**Applies to**

[TFullQueryBuilder](#) component

**Declaration**

```
property QueryBuilders: TQueryBuilders;
```

**Description**

The *QueryBuilders* property is collection of [TQueryBuilder](#) components. Each item in this collection represents single SELECT SQL statement.

---

**See also:**

[AddQureyBuilder method](#)


[DeleteQuqueryBuilder metod](#)




[SetActiveQureyBuilder method](#)

[OnAddQuquery event](#)



## 2.2.3 Methods

 Key methods

-  [AddQueryBuilder](#)
-  [DeleteQueryBuilder](#)
-  [SetActiveQueryBuilder](#)
- [DoAddRemoveLink](#)
- [DoAddRemoveTable](#)
- [BeginReload](#)
- [Clear](#)
- [EndReload](#)

### 2.2.3.1 AddQueryBuilder

**Applies to**

[TFullQueryBuilder](#) component

**Declaration**

```
function AddQueryBuilder(AddUnion: Boolean): TCustomQueryBuilder;
```

**Description**

Call *AddQueryBuilder* method to create new TQueryBuilder component representing single SELECT SQL statement. Set AddUnion parameter to True if you want to add union subquery. Set AddUnion to False if you want to add derived subquery.

---

**See also:**

[QueryBuilders property](#)

[DeleteQueryBuilder metod](#)

[SetActiveQureyBuilder method](#)

[OnAddQuquery event](#)

### 2.2.3.2 DeleteQueryBuilder

**Applies to**

[TFullQueryBuilder](#) component

**Declaration**

```
procedure DeleteQueryBuilder(AName: String);
```

**Description**

Call *DeleteQueryBuilder* method to remove existing TQueryBuilder component representing single SELECT SQL statement. AName parameter is the Name property of removed component.

---

**See also:**

[QueryBuilders property](#)

[AddQureyBuilder method](#)

[SetActiveQureyBuilder method](#)

[OnAddQuuery event](#)

### 2.2.3.3 SetActiveQueryBuilder

**Applies to**

[TFullQueryBuilder](#) component

**Declaration**

```
procedure SetActiveQueryBuilder(AName: String);
```

**Description**

Call *SetActiveQueryBuilder* method to display particular TQueryBuilder component representing single SELECT SQL statement. AName parameter is the Name property of displayed component.

---

**See also:**


[QueryBuilders property](#)


[AddQureyBuilder method](#)

[DeleteQuueryBuilder metod](#)

[OnAddQuuery event](#)

## 2.2.4 Events

 Key events

-  [OnAddQuery](#)
- [OnAddRemoveLink](#)
- [OnAddRemoveTable](#)
- [OnEnterProcParameter](#)
- [OnGetTableFields](#)
- [OnParseError](#)

#### 2.2.4.1 OnAddQuery

**Applies to**

[TFullQueryBuilder](#) component

**Declaration**

```
property OnAddQuery: TNotifyEvent;
```

**Description**

The *OnAddQuery* event takes place when a new TQueryBuilder component is added to the [QueryBuilders](#) collection.

---

**See also:**

[QueryBuilders property](#)

[AddQureyBuilder method](#)

[DeleteQuqueryBuilder metod](#)

[SetActiveQureyBuilder method](#)

## 2.3 TSQLQueryBuilder

### 2.3.1 TSQLQueryBuilder Reference

**Unit**

[QBWindow](#)

**Description**

The *TSQLQueryBuilder* component is intended for creating SELECT statements based on the standard SQL syntax (ANSI 92). It can be used, for example, for working with BDE objects.

## 2.3.2 Properties

▶ Run-time only                       Key properties

- ▶ [Aggregates](#)
- ▶ [Functions](#)
- ▶ [Keywords](#)
- ▶ [Operators](#)
- ▶ [Predicates](#)
- ▶ [Quantifiers](#)
- ▶ [Options](#)
- ▶ [Palette](#)
- ▶ [QuoteIdentifiers](#)
- ▶ [Style](#)
- ▶ [TabHeight](#)
- ▶ [VisibleTabs](#)



### 2.3.2.1 Aggregates

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Aggregates: TStrings;
```

**Description**

The *Aggregates* property contains the list of all the Aggregate functions, available in the 'Grouping criterions' area for use in the query.

---

**See also:**

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.3.2.2 Functions

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Functions: TStrings;
```

**Description**

The *Functions* property contains the list of the SQL functions, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.3.2.3 Keywords

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Keywords: TStrings;
```

**Description**

The *Keywords* property contains the list of all the SQL keywords, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

#### 2.3.2.4 Operators

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Operators: TQBOperators;
```

**Description**

The *Operators* property contains all the operators, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.3.2.5 Options

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBOptions = set of (qboRowSelect, qboDragFieldName, qboHideSelection, qboShowFieldTypes)
```

```
property Options: TQBOptions;
```

#### Description

The *Options* property is a set of suboptions which define the following options of the component:

|                                 |  |
|---------------------------------|--|
| <i>qboRowSelect</i>             | selects the whole row in the 'Criteria' area   |
| <i>qboDragFieldName</i>         | shows the field name on dragging the field   |
| <i>qboHideSelection</i>         | hides the selected items if the object is inactive   |
| <i>qboShowFieldTypes</i>        | shows the field types by hint  |
| <i>qboClearTabsOnTypeChange</i> | indicates to clear 'Criterion' tabs when changing type of the query (SELECT, INSERT, etc.) |

---

#### See also:

[Palette property](#)

### 2.3.2.6 Palette

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

`property` Palette: [TQBPalette](#);

**Description**

The *Palette* property defines various colors, used by the component.

---

**See also:**

[Options property](#)

### 2.3.2.7 QuoteIdentifiers

**Applies to**

[TSQLQueryBuilder](#) component

**Declaration**

```
property QuoteIdentifiers: boolean;
```

**Description**

If property *QuoteIdentifiers* is true, then all the identifiers are quoted when the script is generated.

### 2.3.2.8 Style

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Style: TQBStyle;
```

**Description**

The *Style* property defines the appearance of the 'Criteria' area buttons and 'Builder' area objects.

---

**See also:**

[TQBStyle class](#)



### 2.3.2.9 TabHeight

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property TabHeight: Integer;
```

**Description**

The *TabHeight* property defines the height of the condition panels.

---

**See also:**

[VisibleTabs property](#)

### 2.3.2.10 VisibleTabs

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBVisibleTab = (qbtCriteriaions, qbtSelection, qbtGroupCriteriaions, qbtSorting, qbtData);  
TQBVisibleTabs = set of TQBVisibleTab;
```

```
property VisibleTabs: TQBVisibleTabs;
```

#### Description


The *VisibleTabs* property defines which of the Advanced QueryBuilder tabs ('Criteriaions', 'Selection', Grouping Criteriaions', 'Sorting', 'Data') are visible.






---

#### See also:

[TabHeight property](#)

### 2.3.3 Methods

 Key methods

-  [DoAddRemoveLink](#)
-  [DoAddRemoveTable](#)
-  [BeginReload](#)
-  [Clear](#)
-  [EndReload](#)

### 2.3.3.1 DoAddRemoveLink

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveLink(Link: TQBLink; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveLink* method to add or remove link to the work area. The Link parameter is the TQBLink object, which defines the properties of the link to add or remove. If Action is qbaAdd, then the link, specified by the Link parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveTable method](#)

### 2.3.3.2 DoAddRemoveTable

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveTable(Table: TQBTable; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveTable* method to add or remove table to the work area. The Table parameter is the TQBTable object, which defines the properties of the table to add or remove. If Action is qbaAdd, then the table, specified by the Table parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveLink method](#)

### 2.3.3.3 BeginReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure BeginReload; virtual;
```

**Description**

The *BeginReload* method prepares the 'Criteria' and 'Grouping Criteria' areas for making any changes. You must call this method each time you want to change the [Operators](#) property and call [EndReload](#) when all the changes are done.

---

**See also:**

[Operators property](#)

[EndReload method](#)

#### 2.3.3.4 Clear

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure Clear;
```

**Description**

The *Clear* method clears the work area and all the condition panels.

---

**See also:**

[DoAddRemoveLink method](#)

[DoAddRemoveTable method](#)

### 2.3.3.5 EndReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure EndReload; virtual;
```

**Description**

The *EndReload* method should be called when the changes in the 'Criteria' panel, started by the [BeginReload](#) method are done.

---

**See also:**






[Operators property](#)

[BeginReload method](#)



## 2.3.4 Events

### Key events

-  [OnAddRemoveLink](#)
-  [OnAddRemoveTable](#)
-  [OnEnterProcParameter](#)
-  [OnGetTableFields](#)
-  [OnParserError](#)

### 2.3.4.1 OnAddRemoveLink

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveLinkEvent = procedure(Sender: TObject; Link: TQBLink; Action: TQBAction; var
```

```
property OnAddRemoveLink: TAddRemoveLinkEvent;
```

#### Description

The *OnAddRemoveLink* event takes place when a link is added/removed to/from the component work area. The Link parameter defines the properties of the link to add/remove. Action defines if the link is added or removed. Output parameter Accept allows you to forbid or permit action.

---

#### See also:

[OnAddRemoveTable event](#)

#### 2.3.4.2 OnAddRemoveTable

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveTableEvent = procedure(Sender: TObject; Table: TQBTable; Action: TQBAction
```

```
property OnAddRemoveTable: TAddRemoveTableEvent;
```

##### Description

The *OnAddRemoveTable* event takes place when a table is added/removed to/from the component work area. The Table parameter defines the properties of the table to add/remove. Action defines if the table is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveLink event](#)

### 2.3.4.3 OnEnterProcParameter

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TEnterProcParameter = procedure(Sender: TObject; SProcedure: TQBTable; ListItem: TQ
```

```
property OnEnterProcParameter: TEnterProcParameter;
```

**Description**

The *OnEnterProcParameter* event takes place before the procedure input parameter is added.

---

**See also:**

[OnGetTableFields event](#)

#### 2.3.4.4 OnGetTableFields

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TGetTableFieldsEvent = procedure(const NameOfTable: string; var ADataSet: TDataSet;
```

```
property OnGetTableFields: TGetTableFieldsEvent;
```

**Description**

The *OnGetTableFields* event takes place, when you set the [SQL](#) property in run-time. Constant parameter *NameOfTable* is the table name, used in the SELECT ... FROM statement. Using parameter *ADataSet* you can define the dataset, containing data to select from, and using *AParams* you can define the dataset parameters.

---

**See also:**

[OnEnterProcParameter](#)

#### 2.3.4.5 OnParserError

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property OnParserError: TQBParserErrorEvent;
```

**Description**

The *OnParserError* event takes place when parsing the [SQL property](#) causes an error in the TQBParser object. When error occurs, you receive the following information: error code, line number, line position and token.

---

**See also:**

[Parser property](#)

[SQL property](#)

## 2.4 TInterBaseQueryBuilder

### 2.4.1 TInterBaseQueryBuilder Reference

**Unit**


[QBIBWindow](#)

**Description**

The *TInterBaseQueryBuilder* component is intended for creating queries from InterBase/FireBird tables.

## 2.4.2 Properties

▶ Run-time only

 Key properties

[Aggregates](#)

[Functions](#)

[Keywords](#)

[Operators](#)

[Options](#)

[Palette](#)



[SQLDialect](#)



[SQLServer](#)

[Style](#)

[TabHeight](#)

[VisibleTabs](#)



### 2.4.2.1 Aggregates

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Aggregates: TStrings;
```

**Description**

The *Aggregates* property contains the list of all the Aggregate functions, available in the 'Grouping criterions' area for use in the query.

---

**See also:**

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.4.2.2 Functions

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Functions: TStrings;
```

**Description**

The *Functions* property contains the list of the SQL functions, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.4.2.3 Keywords

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Keywords: TStrings;
```

**Description**

The *Keywords* property contains the list of all the SQL keywords, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

#### 2.4.2.4 Operators

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Operators: TQBOperators;
```

**Description**

The *Operators* property contains all the operators, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Predicates property](#)

[Quantifiers property](#)

#### 2.4.2.5 Options

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBOptions = set of (qboRowSelect, qboDragFieldName, qboHideSelection, qboShowFieldTypes)
```

```
property Options: TQBOptions;
```

##### Description

The *Options* property is a set of suboptions which define the following options of the component:

|                                 |  |
|---------------------------------|--|
| <i>qboRowSelect</i>             | selects the whole row in the 'Criteria' area   |
| <i>qboDragFieldName</i>         | shows the field name on dragging the field   |
| <i>qboHideSelection</i>         | hides the selected items if the object is inactive   |
| <i>qboShowFieldTypes</i>        | shows the field types by hint  |
| <i>qboClearTabsOnTypeChange</i> | indicates to clear 'Criterion' tabs when changing type of the query (SELECT, INSERT, etc.) |

---

##### See also:

[Palette property](#)

#### 2.4.2.6 Palette

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

property Palette: [TQBPalette](#);

**Description**

The *Palette* property defines various colors, used by the component.

---

**See also:**

[Options property](#)

### 2.4.2.7 SQLDialect

**Applies to**

[TInterBaseQueryBuilder](#) component

**Declaration****type**

```
TQBSQLDialect = (qibDialect1, qibDialect3);
```

```
property SQLDialect: TQBSQLDialect;
```

**Description**

The *SQLDialect* property defines, which SQL dialect - Dialect 1 or Dialect 3 should be used.

---

**See also:**

[SQLServer property](#)

#### 2.4.2.8 SQLServer

**Applies to**

[TInterBaseQueryBuilder](#) component

**Declaration****type**

```
TQBIBSQLServer = (qssInterBase, qssFireBird);
```

```
property SQLServer: TQBIBSQLServer;
```

**Description**

The *SQLServer* property defines which database server is used - InterBase or Firebird.

---

**See also:**

[SQLDialect property](#)



#### 2.4.2.9 Style

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Style: TQBStyle;
```

**Description**

The *Style* property defines the appearance of the 'Criteria' area buttons and 'Builder' area objects.

---

**See also:**

[TQBStyle class](#)

#### 2.4.2.10 TabHeight

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property TabHeight: Integer;
```

**Description**

The *TabHeight* property defines the height of the condition panels.

---

**See also:**

[VisibleTabs property](#)

### 2.4.2.11 VisibleTabs

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBVisibleTab = (qbtCriteriaions, qbtSelection, qbtGroupCriteriaions, qbtSorting, qbtData);  
TQBVisibleTabs = set of TQBVisibleTab;
```

```
property VisibleTabs: TQBVisibleTabs;
```

#### Description


The *VisibleTabs* property defines which of the Advanced QueryBuilder tabs ('Criteriaions', 'Selection', Grouping Criteriaions', 'Sorting', 'Data') are visible.






---

#### See also:

[TabHeight property](#)

## 2.4.3 Methods

 Key methods

-  [DoAddRemoveLink](#)
-  [DoAddRemoveTable](#)
-  [BeginReload](#)
-  [Clear](#)
-  [EndReload](#)

### 2.4.3.1 DoAddRemoveLink

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveLink(Link: TQBLink; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveLink* method to add or remove link to the work area. The Link parameter is the TQBLink object, which defines the properties of the link to add or remove. If Action is qbaAdd, then the link, specified by the Link parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveTable method](#)

### 2.4.3.2 DoAddRemoveTable

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveTable(Table: TQBTable; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveTable* method to add or remove table to the work area. The Table parameter is the TQBTable object, which defines the properties of the table to add or remove. If Action is qbaAdd, then the table, specified by the Table parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveLink method](#)

### 2.4.3.3 BeginReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure BeginReload; virtual;
```

**Description**

The *BeginReload* method prepares the 'Criteria' and 'Grouping Criteria' areas for making any changes. You must call this method each time you want to change the [Operators](#) property and call [EndReload](#) when all the changes are done.

---

**See also:**

[Operators property](#)

[EndReload method](#)

#### 2.4.3.4 Clear

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure Clear;
```

**Description**

The *Clear* method clears the work area and all the condition panels.

---

**See also:**

[DoAddRemoveLink method](#)

[DoAddRemoveTable method](#)



#### 2.4.3.5 EndReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure EndReload; virtual;
```

**Description**

The *EndReload* method should be called when the changes in the 'Criteria' panel, started by the [BeginReload](#) method are done.

---






**See also:**

[Operators property](#)

[BeginReload method](#)

## 2.4.4 Events

### Key events

-  [OnAddRemoveLink](#)
-  [OnAddRemoveTable](#)
-  [OnEnterProcParameter](#)
-  [OnGetTableFields](#)
-  [OnParserError](#)

#### 2.4.4.1 OnAddRemoveLink

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveLinkEvent = procedure(Sender: TObject; Link: TQBLink; Action: TQBAction; var
```

```
property OnAddRemoveLink: TAddRemoveLinkEvent;
```

##### Description

The *OnAddRemoveLink* event takes place when a link is added/removed to/from the component work area. The Link parameter defines the properties of the link to add/remove. Action defines if the link is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveTable event](#)

#### 2.4.4.2 OnAddRemoveTable

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveTableEvent = procedure(Sender: TObject; Table: TQBTable; Action: TQBAction
```

```
property OnAddRemoveTable: TAddRemoveTableEvent;
```

##### Description

The *OnAddRemoveTable* event takes place when a table is added/removed to/from the component work area. The Table parameter defines the properties of the table to add/remove. Action defines if the table is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveLink event](#)

### 2.4.4.3 OnEnterProcParameter

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TEnterProcParameter = procedure(Sender: TObject; SProcedure: TQBTable; ListItem: TQ
```

```
property OnEnterProcParameter: TEnterProcParameter;
```

**Description**

The *OnEnterProcParameter* event takes place before the procedure input parameter is added.

---

**See also:**

[OnGetTableFields event](#)

#### 2.4.4.4 OnGetTableFields

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TGetTableFieldsEvent = procedure(const NameOfTable: string; var ADataSet: TDataSet;
```

```
property OnGetTableFields: TGetTableFieldsEvent;
```

**Description**

The *OnGetTableFields* event takes place, when you set the [SQL](#) property in run-time. Constant parameter *NameOfTable* is the table name, used in the SELECT ... FROM statement. Using parameter *ADataSet* you can define the dataset, containing data to select from, and using *AParams* you can define the dataset parameters.

---

**See also:**

[OnEnterProcParameter](#)

#### 2.4.4.5 OnParserError

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property OnParserError: TQBParserErrorEvent;
```

**Description**

The *OnParserError* event takes place when parsing the [SQL property](#) causes an error in the TQBParser object. When error occurs, you receive the following information: error code, line number, line position and token.

---

**See also:**

[Parser property](#)

[SQL property](#)

## 2.5 TMySQLQueryBuilder

### 2.5.1 TMySQLQueryBuilder Reference

**Unit**

[QMyWindow](#)


**Description**

The *TMySQLQueryBuilder* component is used for creating queries from the MySQL tables.



## 2.5.2 Properties

▶ Run-time only

 Key properties

[Aggregates](#)

[Functions](#)

[Keywords](#)

[Operators](#)

[Options](#)

[Palette](#)

[QuoteIdentifiers](#)

[Style](#)

[TabHeight](#)

[VisibleTabs](#)

### 2.5.2.1 Aggregates

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Aggregates: TStrings;
```

**Description**

The *Aggregates* property contains the list of all the Aggregate functions, available in the 'Grouping criterions' area for use in the query.

---

**See also:**

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.5.2.2 Functions

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Functions: TStrings;
```

**Description**

The *Functions* property contains the list of the SQL functions, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.5.2.3 Keywords

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Keywords: TStrings;
```

**Description**

The *Keywords* property contains the list of all the SQL keywords, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

#### 2.5.2.4 Operators

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Operators: TQBOperators;
```

**Description**

The *Operators* property contains all the operators, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.5.2.5 Options

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBOptions = set of (qboRowSelect, qboDragFieldName, qboHideSelection, qboShowFieldTypes)
```

```
property Options: TQBOptions;
```

#### Description

The *Options* property is a set of suboptions which define the following options of the component:

|                                 |  |
|---------------------------------|--|
| <i>qboRowSelect</i>             | selects the whole row in the 'Criteria' area   |
| <i>qboDragFieldName</i>         | shows the field name on dragging the field   |
| <i>qboHideSelection</i>         | hides the selected items if the object is inactive   |
| <i>qboShowFieldTypes</i>        | shows the field types by hint  |
| <i>qboClearTabsOnTypeChange</i> | indicates to clear 'Criterion' tabs when changing type of the query (SELECT, INSERT, etc.) |

---

#### See also:

[Palette property](#)

### 2.5.2.6 Palette

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

property Palette: [TQBPalette](#);

**Description**

The *Palette* property defines various colors, used by the component.

---

**See also:**

[Options property](#)

### 2.5.2.7 QuoteIdentifiers

**Applies to**

[TSQLQueryBuilder](#) component

**Declaration**

```
property QuoteIdentifiers: boolean;
```

**Description**

If property *QuoteIdentifiers* is true, then all the identifiers are quoted when the script is generated.



### 2.5.2.8 Style

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Style: TQBStyle;
```

**Description**

The *Style* property defines the appearance of the 'Criteria' area buttons and 'Builder' area objects.

---

**See also:**

[TQBStyle class](#)

### 2.5.2.9 TabHeight

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property TabHeight: Integer;
```

**Description**

The *TabHeight* property defines the height of the condition panels.

---

**See also:**

[VisibleTabs property](#)

### 2.5.2.10 VisibleTabs

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBVisibleTab = (qbtCriteriaions, qbtSelection, qbtGroupCriteriaions, qbtSorting, qbtData);  
TQBVisibleTabs = set of TQBVisibleTab;
```

```
property VisibleTabs: TQBVisibleTabs;
```

**Description**


The *VisibleTabs* property defines which of the Advanced QueryBuilder tabs ('Criteriaions', 'Selection', Grouping Criteriaions', 'Sorting', 'Data') are visible.






---

**See also:**

[TabHeight property](#)

## 2.5.3 Methods

 Key methods

-  [DoAddRemoveLink](#)
-  [DoAddRemoveTable](#)
-  [BeginReload](#)
-  [Clear](#)
-  [EndReload](#)

### 2.5.3.1 DoAddRemoveLink

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveLink(Link: TQBLink; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveLink* method to add or remove link to the work area. The Link parameter is the TQBLink object, which defines the properties of the link to add or remove. If Action is qbaAdd, then the link, specified by the Link parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveTable method](#)

### 2.5.3.2 DoAddRemoveTable

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveTable(Table: TQBTable; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveTable* method to add or remove table to the work area. The Table parameter is the TQBTable object, which defines the properties of the table to add or remove. If Action is qbaAdd, then the table, specified by the Table parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveLink method](#)

### 2.5.3.3 BeginReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure BeginReload; virtual;
```

**Description**

The *BeginReload* method prepares the 'Criteria' and 'Grouping Criteria' areas for making any changes. You must call this method each time you want to change the [Operators](#) property and call [EndReload](#) when all the changes are done.

---

**See also:**

[Operators property](#)

[EndReload method](#)

#### 2.5.3.4 Clear

**Applies to**  
[TCustomQueryBuilder](#) component

**Declaration**  
`procedure Clear;`

**Description**

The *Clear* method clears the work area and all the condition panels.

---

**See also:**

[DoAddRemoveLink method](#)  
[DoAddRemoveTable method](#)



### 2.5.3.5 EndReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure EndReload; virtual;
```

**Description**

The *EndReload* method should be called when the changes in the 'Criteria' panel, started by the [BeginReload](#) method are done.

---






**See also:**

[Operators property](#)

[BeginReload method](#)

## 2.5.4 Events

### Key events

-  [OnAddRemoveLink](#)
-  [OnAddRemoveTable](#)
-  [OnEnterProcParameter](#)
-  [OnGetTableFields](#)
-  [OnParserError](#)

#### 2.5.4.1 OnAddRemoveLink

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveLinkEvent = procedure(Sender: TObject; Link: TQBLink; Action: TQBAction; v
```

```
property OnAddRemoveLink: TAddRemoveLinkEvent;
```

**Description**

The *OnAddRemoveLink* event takes place when a link is added/removed to/from the component work area. The Link parameter defines the properties of the link to add/remove. Action defines if the link is added or removed. Output parameter Accept allows you to forbid or permit action.

---

**See also:**

[OnAddRemoveTable event](#)

#### 2.5.4.2 OnAddRemoveTable

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveTableEvent = procedure(Sender: TObject; Table: TQBTable; Action: TQBAction
```

```
property OnAddRemoveTable: TAddRemoveTableEvent;
```

##### Description

The *OnAddRemoveTable* event takes place when a table is added/removed to/from the component work area. The Table parameter defines the properties of the table to add/remove. Action defines if the table is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveLink event](#)

### 2.5.4.3 OnEnterProcParameter

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TEnterProcParameter = procedure(Sender: TObject; SProcedure: TQBTable; ListItem: TQ
```

```
property OnEnterProcParameter: TEnterProcParameter;
```

**Description**

The *OnEnterProcParameter* event takes place before the procedure input parameter is added.

---

**See also:**

[OnGetTableFields event](#)

#### 2.5.4.4 OnGetTableFields

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TGetTableFieldsEvent = procedure(const NameOfTable: string; var ADataSet: TDataSet;
```

```
property OnGetTableFields: TGetTableFieldsEvent;
```

**Description**

The *OnGetTableFields* event takes place, when you set the [SQL](#) property in run-time. Constant parameter *NameOfTable* is the table name, used in the SELECT ... FROM statement. Using parameter *ADataSet* you can define the dataset, containing data to select from, and using *AParams* you can define the dataset parameters.

---

**See also:**

[OnEnterProcParameter](#)

#### 2.5.4.5 OnParserError

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property OnParserError: TQBParserErrorEvent;
```

**Description**

The *OnParserError* event takes place when parsing the [SQL property](#) causes an error in the TQBParser object. When error occurs, you receive the following information: error code, line number, line position and token.

---

**See also:**

[Parser property](#)

[SQL property](#)

## 2.6 TPgSQLQueryBuilder

### 2.6.1 TPgSQLQueryBuilder Reference

**Unit**

[QBPgWindow](#)


**Description**

The *TPgSQLQueryBuilder* component is used for creating queries from the PostgreSQL tables.



## 2.6.2 Properties

▶ Run-time only

 Key properties

[Aggregates](#)

[Functions](#)

[Keywords](#)

[Operators](#)

[Options](#)

[Palette](#)

[Style](#)

[TabHeight](#)



[UseNamespaces](#)

[VisibleTabs](#)

### 2.6.2.1 Aggregates

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Aggregates: TStrings;
```

**Description**

The *Aggregates* property contains the list of all the Aggregate functions, available in the 'Grouping criterions' area for use in the query.

---

**See also:**

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.6.2.2 Functions

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Functions: TStrings;
```

**Description**

The *Functions* property contains the list of the SQL functions, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.6.2.3 Keywords

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Keywords: TStrings;
```

**Description**

The *Keywords* property contains the list of all the SQL keywords, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

#### 2.6.2.4 Operators

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Operators: TQBOperators;
```

**Description**

The *Operators* property contains all the operators, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.6.2.5 Options

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBOptions = set of (qboRowSelect, qboDragFieldName, qboHideSelection, qboShowFieldTypes)
```

```
property Options: TQBOptions;
```

#### Description

The *Options* property is a set of suboptions which define the following options of the component:

|                                 |  |
|---------------------------------|--|
| <i>qboRowSelect</i>             | selects the whole row in the 'Criteria' area   |
| <i>qboDragFieldName</i>         | shows the field name on dragging the field   |
| <i>qboHideSelection</i>         | hides the selected items if the object is inactive   |
| <i>qboShowFieldTypes</i>        | shows the field types by hint  |
| <i>qboClearTabsOnTypeChange</i> | indicates to clear 'Criterion' tabs when changing type of the query (SELECT, INSERT, etc.) |

---

#### See also:

[Palette property](#)

#### 2.6.2.6 Palette

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

property Palette: [TQBPalette](#);

**Description**

The *Palette* property defines various colors, used by the component.

---

**See also:**

[Options property](#)

### 2.6.2.7 Style

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Style: TQBStyle;
```

**Description**

The *Style* property defines the appearance of the 'Criteria's' area buttons and 'Builder' area objects.

---

**See also:**

[TQBStyle class](#)



### 2.6.2.8 TabHeight

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property TabHeight: Integer;
```

**Description**

The *TabHeight* property defines the height of the condition panels.

---

**See also:**

[VisibleTabs property](#)

### 2.6.2.9 UseNamespaces

**Applies to**

[TPgSQLQueryBuilder](#) component

**Declaration**

```
property UseNamespaces: boolean;
```

**Description**

If the *UseNamespaces* property is true, namespaces are used in building and parsing SQL statements. Use this for working with PostgreSQL v7.3 and higher.

### 2.6.2.10 VisibleTabs

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBVisibleTab = (qbtCriteriaions, qbtSelection, qbtGroupCriteriaions, qbtSorting, qbtData);  
TQBVisibleTabs = set of TQBVisibleTab;
```

```
property VisibleTabs: TQBVisibleTabs;
```

#### Description


The *VisibleTabs* property defines which of the Advanced QueryBuilder tabs ('Criteriaions', 'Selection', Grouping Criteriaions', 'Sorting', 'Data') are visible.






---

#### See also:

[TabHeight property](#)

## 2.6.3 Methods

 Key methods

-  [DoAddRemoveLink](#)
-  [DoAddRemoveTable](#)
-  [BeginReload](#)
-  [Clear](#)
-  [EndReload](#)

### 2.6.3.1 DoAddRemoveLink

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveLink(Link: TQBLink; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveLink* method to add or remove link to the work area. The Link parameter is the TQBLink object, which defines the properties of the link to add or remove. If Action is qbaAdd, then the link, specified by the Link parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveTable method](#)

### 2.6.3.2 DoAddRemoveTable

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveTable(Table: TQBTable; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveTable* method to add or remove table to the work area. The Table parameter is the TQBTable object, which defines the properties of the table to add or remove. If Action is qbaAdd, then the table, specified by the Table parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveLink method](#)

### 2.6.3.3 BeginReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure BeginReload; virtual;
```

**Description**

The *BeginReload* method prepares the 'Criteria' and 'Grouping Criteria' areas for making any changes. You must call this method each time you want to change the [Operators](#) property and call [EndReload](#) when all the changes are done.

---

**See also:**

[Operators property](#)

[EndReload method](#)

#### 2.6.3.4 Clear

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure Clear;
```

**Description**

The *Clear* method clears the work area and all the condition panels.

---

**See also:**

[DoAddRemoveLink method](#)

[DoAddRemoveTable method](#)



### 2.6.3.5 EndReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure EndReload; virtual;
```

**Description**

The *EndReload* method should be called when the changes in the 'Criteria' panel, started by the [BeginReload](#) method are done.


---

**See also:**

[Operators property](#)

[BeginReload method](#)

## 2.6.4 Events

 Key events

[OnAddRemoveLink](#)  
[OnAddRemoveTable](#)  
[OnEnterProcParameter](#)  
[OnGetTableFields](#)  
[OnParserError](#)

#### 2.6.4.1 OnAddRemoveLink

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveLinkEvent = procedure(Sender: TObject; Link: TQBLink; Action: TQBAction; var
```

```
property OnAddRemoveLink: TAddRemoveLinkEvent;
```

##### Description

The *OnAddRemoveLink* event takes place when a link is added/removed to/from the component work area. The Link parameter defines the properties of the link to add/remove. Action defines if the link is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveTable event](#)

#### 2.6.4.2 OnAddRemoveTable

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveTableEvent = procedure(Sender: TObject; Table: TQBTable; Action: TQBAction
```

```
property OnAddRemoveTable: TAddRemoveTableEvent;
```

##### Description

The *OnAddRemoveTable* event takes place when a table is added/removed to/from the component work area. The Table parameter defines the properties of the table to add/remove. Action defines if the table is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveLink event](#)

### 2.6.4.3 OnEnterProcParameter

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TEnterProcParameter = procedure(Sender: TObject; SProcedure: TQBTable; ListItem: TQ
```

```
property OnEnterProcParameter: TEnterProcParameter;
```

**Description**

The *OnEnterProcParameter* event takes place before the procedure input parameter is added.

---

**See also:**

[OnGetTableFields event](#)

#### 2.6.4.4 OnGetTableFields

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TGetTableFieldsEvent = procedure(const NameOfTable: string; var ADataSet: TDataSet;
```

```
property OnGetTableFields: TGetTableFieldsEvent;
```

**Description**

The *OnGetTableFields* event takes place, when you set the [SQL](#) property in run-time. Constant parameter *NameOfTable* is the table name, used in the SELECT ... FROM statement. Using parameter *ADataSet* you can define the dataset, containing data to select from, and using *AParams* you can define the dataset parameters.

---

**See also:**

[OnEnterProcParameter](#)

#### 2.6.4.5 OnParserError

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property OnParserError: TQBParserErrorEvent;
```

**Description**

The *OnParserError* event takes place when parsing the [SQL property](#) causes an error in the TQBParser object. When error occurs, you receive the following information: error code, line number, line position and token.

---

**See also:**

[Parser property](#)

[SQL property](#)

## 2.7 TMSSQLQueryBuilder

### 2.7.1 TMSSQLQueryBuilder Reference

**Unit**

[QBMSWindow](#)

**Description**

The *TMSSQLQueryBuilder* component is used for creating queries from the Microsoft SQL tables.



## 2.7.2 Properties

▶ Run-time only

🔑 Key properties

🔑 [Aggregates](#)  
[AlterQuotes](#)  
[Functions](#)  
[Keywords](#)  
[Operators](#)  
[Options](#)  
[Palette](#)  
[Style](#)  
[TabHeight](#)  
[VisibleTabs](#)

### 2.7.2.1 Aggregates

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Aggregates: TStrings;
```

**Description**

The *Aggregates* property contains the list of all the Aggregate functions, available in the 'Grouping criterions' area for use in the query.

---

**See also:**

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.7.2.2 AlterQuotes

**Applies to**

[TMSSQLQueryBuilder](#) component

**Declaration**

```
property AlterQuotes: boolean;
```

**Description**

If the *AlterQuotes* property is True, then the "" characters are used for quoting identifiers instead of [] characters.

### 2.7.2.3 Functions

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Functions: TStrings;
```

**Description**

The *Functions* property contains the list of the SQL functions, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

#### 2.7.2.4 Keywords

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Keywords: TStrings;
```

**Description**

The *Keywords* property contains the list of all the SQL keywords, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.7.2.5 Operators

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Operators: TQBOperators;
```

**Description**

The *Operators* property contains all the operators, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.7.2.6 Options

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBOptions = set of (qboRowSelect, qboDragFieldName, qboHideSelection, qboShowFieldTypes)
```

```
property Options: TQBOptions;
```

#### Description

The *Options* property is a set of suboptions which define the following options of the component:

|                                 |  |
|---------------------------------|--|
| <i>qboRowSelect</i>             | selects the whole row in the 'Criteria' area   |
| <i>qboDragFieldName</i>         | shows the field name on dragging the field   |
| <i>qboHideSelection</i>         | hides the selected items if the object is inactive   |
| <i>qboShowFieldTypes</i>        | shows the field types by hint  |
| <i>qboClearTabsOnTypeChange</i> | indicates to clear 'Criterion' tabs when changing type of the query (SELECT, INSERT, etc.) |

---

#### See also:

[Palette property](#)

### 2.7.2.7 Palette

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

`property` Palette: [TQBPalette](#);

**Description**

The *Palette* property defines various colors, used by the component.

---

**See also:**

[Options property](#)



### 2.7.2.8 Style

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

property Style: [TQBStyle](#);

**Description**

The *Style* property defines the appearance of the 'Criteria' area buttons and 'Builder' area objects.

---

**See also:**

[TQBStyle class](#)

### 2.7.2.9 TabHeight

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property TabHeight: Integer;
```

**Description**

The *TabHeight* property defines the height of the condition panels.

---

**See also:**

[VisibleTabs property](#)

### 2.7.2.10 VisibleTabs

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBVisibleTab = (qbtCriteriaions, qbtSelection, qbtGroupCriteriaions, qbtSorting, qbtData);  
TQBVisibleTabs = set of TQBVisibleTab;
```

```
property VisibleTabs: TQBVisibleTabs;
```

**Description**


The *VisibleTabs* property defines which of the Advanced QueryBuilder tabs ('Criteriaions', 'Selection', Grouping Criteriaions', 'Sorting', 'Data') are visible.






---

**See also:**

[TabHeight property](#)

## 2.7.3 Methods

 Key methods

-  [DoAddRemoveLink](#)
-  [DoAddRemoveTable](#)
-  [BeginReload](#)
-  [Clear](#)
-  [EndReload](#)

### 2.7.3.1 DoAddRemoveLink

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveLink(Link: TQBLink; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveLink* method to add or remove link to the work area. The Link parameter is the TQBLink object, which defines the properties of the link to add or remove. If Action is qbaAdd, then the link, specified by the Link parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveTable method](#)

### 2.7.3.2 DoAddRemoveTable

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveTable(Table: TQBTable; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveTable* method to add or remove table to the work area. The Table parameter is the TQBTable object, which defines the properties of the table to add or remove. If Action is qbaAdd, then the table, specified by the Table parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveLink method](#)

### 2.7.3.3 BeginReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure BeginReload; virtual;
```

**Description**

The *BeginReload* method prepares the 'Criteria' and 'Grouping Criteria' areas for making any changes. You must call this method each time you want to change the [Operators](#) property and call [EndReload](#) when all the changes are done.

---

**See also:**

[Operators property](#)

[EndReload method](#)

#### 2.7.3.4 Clear

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure Clear;
```

**Description**

The *Clear* method clears the work area and all the condition panels.

---

**See also:**

[DoAddRemoveLink method](#)

[DoAddRemoveTable method](#)



### 2.7.3.5 EndReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure EndReload; virtual;
```

**Description**

The *EndReload* method should be called when the changes in the 'Criteria' panel, started by the [BeginReload](#) method are done.


---

**See also:**

[Operators property](#)

[BeginReload method](#)

## 2.7.4 Events

 Key events

[OnAddRemoveLink](#)  
[OnAddRemoveTable](#)  
[OnEnterProcParameter](#)  
[OnGetTableFields](#)  
[OnParserError](#)

#### 2.7.4.1 OnAddRemoveLink

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveLinkEvent = procedure(Sender: TObject; Link: TQBLink; Action: TQBAction; var
```

```
property OnAddRemoveLink: TAddRemoveLinkEvent;
```

##### Description

The *OnAddRemoveLink* event takes place when a link is added/removed to/from the component work area. The Link parameter defines the properties of the link to add/remove. Action defines if the link is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveTable event](#)

#### 2.7.4.2 OnAddRemoveTable

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveTableEvent = procedure(Sender: TObject; Table: TQBTable; Action: TQBAction
```

```
property OnAddRemoveTable: TAddRemoveTableEvent;
```

##### Description

The *OnAddRemoveTable* event takes place when a table is added/removed to/from the component work area. The Table parameter defines the properties of the table to add/remove. Action defines if the table is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveLink event](#)

### 2.7.4.3 OnEnterProcParameter

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TEnterProcParameter = procedure(Sender: TObject; SProcedure: TQBTable; ListItem: TQ
```

```
property OnEnterProcParameter: TEnterProcParameter;
```

**Description**

The *OnEnterProcParameter* event takes place before the procedure input parameter is added.

---

**See also:**

[OnGetTableFields event](#)

#### 2.7.4.4 OnGetTableFields

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TGetTableFieldsEvent = procedure(const NameOfTable: string; var ADataSet: TDataSet;
```

```
property OnGetTableFields: TGetTableFieldsEvent;
```

**Description**

The *OnGetTableFields* event takes place, when you set the [SQL](#) property in run-time. Constant parameter *NameOfTable* is the table name, used in the SELECT ... FROM statement. Using parameter *ADataSet* you can define the dataset, containing data to select from, and using *AParams* you can define the dataset parameters.

---

**See also:**

[OnEnterProcParameter](#)

#### 2.7.4.5 OnParserError

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property OnParserError: TQBParserErrorEvent;
```

**Description**

The *OnParserError* event takes place when parsing the [SQL property](#) causes an error in the TQBParser object. When error occurs, you receive the following information: error code, line number, line position and token.

---

**See also:**

[Parser property](#)

[SQL property](#)

## 2.8 TDb2QueryBuilder

### 2.8.1 TDb2QueryBuilder Reference

**Unit**

[QDb2Window](#)


**Description**

The *TDb2QueryBuilder* component is used for creating queries from the IBM DB2 tables.



## 2.8.2 Properties

▶ Run-time only

 Key properties

[Aggregates](#)

[Functions](#)

[Keywords](#)

[Operators](#)

[Options](#)

[Palette](#)

[Style](#)

[TabHeight](#)



[UseNamespaces](#)

[VisibleTabs](#)

### 2.8.2.1 Aggregates

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Aggregates: TStrings;
```

**Description**

The *Aggregates* property contains the list of all the Aggregate functions, available in the 'Grouping criterions' area for use in the query.

---

**See also:**

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.8.2.2 Functions

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Functions: TStrings;
```

**Description**

The *Functions* property contains the list of the SQL functions, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.8.2.3 Keywords

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Keywords: TStrings;
```

**Description**

The *Keywords* property contains the list of all the SQL keywords, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

#### 2.8.2.4 Operators

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Operators: TQBOperators;
```

**Description**

The *Operators* property contains all the operators, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.8.2.5 Options

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBOptions = set of (qboRowSelect, qboDragFieldName, qboHideSelection, qboShowFieldTypes)
```

```
property Options: TQBOptions;
```

#### Description

The *Options* property is a set of suboptions which define the following options of the component:

|                                 |  |
|---------------------------------|--|
| <i>qboRowSelect</i>             | selects the whole row in the 'Criteria' area   |
| <i>qboDragFieldName</i>         | shows the field name on dragging the field   |
| <i>qboHideSelection</i>         | hides the selected items if the object is inactive   |
| <i>qboShowFieldTypes</i>        | shows the field types by hint  |
| <i>qboClearTabsOnTypeChange</i> | indicates to clear 'Criterion' tabs when changing type of the query (SELECT, INSERT, etc.) |

---

#### See also:

[Palette property](#)

### 2.8.2.6 Palette

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

property Palette: [TQBPalette](#);

**Description**

The *Palette* property defines various colors, used by the component.

---

**See also:**

[Options property](#)

### 2.8.2.7 Style

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Style: TQBStyle;
```

**Description**

The *Style* property defines the appearance of the 'Criteria's' area buttons and 'Builder' area objects.

---

**See also:**

[TQBStyle class](#)



### 2.8.2.8 TabHeight

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property TabHeight: Integer;
```

**Description**

The *TabHeight* property defines the height of the condition panels.

---

**See also:**

[VisibleTabs property](#)

### 2.8.2.9 UseNamespaces

**Applies to**

[TDb2QueryBuilder](#) component

**Declaration**

```
property UseNamespaces: boolean;
```

**Description**

If the *UseNamespaces* property is true, namespaces are used in building and parsing SQL statements.

### 2.8.2.10 VisibleTabs

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBVisibleTab = (qbtCriteriaions, qbtSelection, qbtGroupCriteriaions, qbtSorting, qbtData);  
TQBVisibleTabs = set of TQBVisibleTab;
```

```
property VisibleTabs: TQBVisibleTabs;
```

**Description**


The *VisibleTabs* property defines which of the Advanced QueryBuilder tabs ('Criteriaions', 'Selection', Grouping Criteriaions', 'Sorting', 'Data') are visible.






---

**See also:**

[TabHeight property](#)

## 2.8.3 Methods

 Key methods

-  [DoAddRemoveLink](#)
-  [DoAddRemoveTable](#)
-  [BeginReload](#)
-  [Clear](#)
-  [EndReload](#)

### 2.8.3.1 DoAddRemoveLink

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveLink(Link: TQBLink; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveLink* method to add or remove link to the work area. The Link parameter is the TQBLink object, which defines the properties of the link to add or remove. If Action is qbaAdd, then the link, specified by the Link parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveTable method](#)

### 2.8.3.2 DoAddRemoveTable

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveTable(Table: TQBTable; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveTable* method to add or remove table to the work area. The Table parameter is the TQBTable object, which defines the properties of the table to add or remove. If Action is qbaAdd, then the table, specified by the Table parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveLink method](#)

### 2.8.3.3 BeginReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure BeginReload; virtual;
```

**Description**

The *BeginReload* method prepares the 'Criteria' and 'Grouping Criteria' areas for making any changes. You must call this method each time you want to change the [Operators](#) property and call [EndReload](#) when all the changes are done.

---

**See also:**

[Operators property](#)

[EndReload method](#)

#### 2.8.3.4 Clear

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure Clear;
```

**Description**

The *Clear* method clears the work area and all the condition panels.

---

**See also:**

[DoAddRemoveLink method](#)

[DoAddRemoveTable method](#)



### 2.8.3.5 EndReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure EndReload; virtual;
```

**Description**

The *EndReload* method should be called when the changes in the 'Criteria' panel, started by the [BeginReload](#) method are done.


---

**See also:**

[Operators property](#)

[BeginReload method](#)

## 2.8.4 Events

 Key events

[OnAddRemoveLink](#)  
[OnAddRemoveTable](#)  
[OnEnterProcParameter](#)  
[OnGetTableFields](#)  
[OnParserError](#)

#### 2.8.4.1 OnAddRemoveLink

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveLinkEvent = procedure(Sender: TObject; Link: TQBLink; Action: TQBAction; var
```

```
property OnAddRemoveLink: TAddRemoveLinkEvent;
```

##### Description

The *OnAddRemoveLink* event takes place when a link is added/removed to/from the component work area. The Link parameter defines the properties of the link to add/remove. Action defines if the link is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveTable event](#)

#### 2.8.4.2 OnAddRemoveTable

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveTableEvent = procedure(Sender: TObject; Table: TQBTable; Action: TQBAction
```

```
property OnAddRemoveTable: TAddRemoveTableEvent;
```

##### Description

The *OnAddRemoveTable* event takes place when a table is added/removed to/from the component work area. The Table parameter defines the properties of the table to add/remove. Action defines if the table is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveLink event](#)

### 2.8.4.3 OnEnterProcParameter

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TEnterProcParameter = procedure(Sender: TObject; SProcedure: TQBTable; ListItem: TQ
```

```
property OnEnterProcParameter: TEnterProcParameter;
```

**Description**

The *OnEnterProcParameter* event takes place before the procedure input parameter is added.

---

**See also:**

[OnGetTableFields event](#)

#### 2.8.4.4 OnGetTableFields

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TGetTableFieldsEvent = procedure(const NameOfTable: string; var ADataSet: TDataSet;
```

```
property OnGetTableFields: TGetTableFieldsEvent;
```

**Description**

The *OnGetTableFields* event takes place, when you set the [SQL](#) property in run-time. Constant parameter *NameOfTable* is the table name, used in the SELECT ... FROM statement. Using parameter *ADataSet* you can define the dataset, containing data to select from, and using *AParams* you can define the dataset parameters.

---

**See also:**

[OnEnterProcParameter](#)

#### 2.8.4.5 OnParserError

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property OnParserError: TQBParserErrorEvent;
```

**Description**

The *OnParserError* event takes place when parsing the [SQL property](#) causes an error in the TQBParser object. When error occurs, you receive the following information: error code, line number, line position and token.

---

**See also:**

[Parser property](#)

[SQL property](#)

## 2.9 TDbiQueryBuilder

### 2.9.1 TDbiQueryBuilder Reference

**Unit**

[QBDbiWindow](#)


**Description**

The *TDbiQueryBuilder* component is used for creating queries from the DBISAM tables.



## 2.9.2 Properties

▶ Run-time only

 Key properties

[Aggregates](#)

[Functions](#)

[Keywords](#)

[Operators](#)

[Options](#)

[Palette](#)

[Style](#)

[TabHeight](#)

[VisibleTabs](#)

### 2.9.2.1 Aggregates

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Aggregates: TStrings;
```

**Description**

The *Aggregates* property contains the list of all the Aggregate functions, available in the 'Grouping criterions' area for use in the query.

---

**See also:**

[Functions property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.9.2.2 Functions

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Functions: TStrings;
```

**Description**

The *Functions* property contains the list of the SQL functions, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Keywords property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.9.2.3 Keywords

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Keywords: TStrings;
```

**Description**

The *Keywords* property contains the list of all the SQL keywords, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Operators property](#)

[Predicates property](#)

[Quantifiers property](#)

#### 2.9.2.4 Operators

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Operators: TQBOperators;
```

**Description**

The *Operators* property contains all the operators, available for use in the query.

---

**See also:**

[Aggregates property](#)

[Functions property](#)

[Keywords property](#)

[Predicates property](#)

[Quantifiers property](#)

### 2.9.2.5 Options

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBOptions = set of (qboRowSelect, qboDragFieldName, qboHideSelection, qboShowFieldTypes)
```

```
property Options: TQBOptions;
```

#### Description

The *Options* property is a set of suboptions which define the following options of the component:

|                                 |  |
|---------------------------------|--|
| <i>qboRowSelect</i>             | selects the whole row in the 'Criteria' area   |
| <i>qboDragFieldName</i>         | shows the field name on dragging the field   |
| <i>qboHideSelection</i>         | hides the selected items if the object is inactive   |
| <i>qboShowFieldTypes</i>        | shows the field types by hint  |
| <i>qboClearTabsOnTypeChange</i> | indicates to clear 'Criterion' tabs when changing type of the query (SELECT, INSERT, etc.) |

---

#### See also:

[Palette property](#)

### 2.9.2.6 Palette

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

property Palette: [TQBPalette](#);

**Description**

The *Palette* property defines various colors, used by the component.

---

**See also:**

[Options property](#)

### 2.9.2.7 Style

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property Style: TQBStyle;
```

**Description**

The *Style* property defines the appearance of the 'Criteria' area buttons and 'Builder' area objects.

---

**See also:**

[TQBStyle class](#)



### 2.9.2.8 TabHeight

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property TabHeight: Integer;
```

**Description**

The *TabHeight* property defines the height of the condition panels.

---

**See also:**

[VisibleTabs property](#)

### 2.9.2.9 VisibleTabs

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBVisibleTab = (qbtCriteriaions, qbtSelection, qbtGroupCriteriaions, qbtSorting, qbtData);  
TQBVisibleTabs = set of TQBVisibleTab;
```

```
property VisibleTabs: TQBVisibleTabs;
```

#### Description


The *VisibleTabs* property defines which of the Advanced QueryBuilder tabs ('Criteriaions', 'Selection', Grouping Criteriaions', 'Sorting', 'Data') are visible.






---

#### See also:

[TabHeight property](#)

## 2.9.3 Methods

 Key methods

-  [DoAddRemoveLink](#)
-  [DoAddRemoveTable](#)
-  [BeginReload](#)
-  [Clear](#)
-  [EndReload](#)

### 2.9.3.1 DoAddRemoveLink

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveLink(Link: TQBLink; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveLink* method to add or remove link to the work area. The Link parameter is the TQBLink object, which defines the properties of the link to add or remove. If Action is qbaAdd, then the link, specified by the Link parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveTable method](#)

### 2.9.3.2 DoAddRemoveTable

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TQBAction = (qbaAdd, qbaRemove);
```

```
function DoAddRemoveTable(Table: TQBTable; Action: TQBAction): Boolean;
```

**Description**

Use the *DoAddRemoveTable* method to add or remove table to the work area. The Table parameter is the TQBTable object, which defines the properties of the table to add or remove. If Action is qbaAdd, then the table, specified by the Table parameter, is being added to the work area, otherwise it's being removed.

---

**See also:**

[DoAddRemoveLink method](#)

### 2.9.3.3 BeginReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure BeginReload; virtual;
```

**Description**

The *BeginReload* method prepares the 'Criteria' and 'Grouping Criteria' areas for making any changes. You must call this method each time you want to change the [Operators](#) property and call [EndReload](#) when all the changes are done.

---

**See also:**

[Operators property](#)

[EndReload method](#)

#### 2.9.3.4 Clear

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure Clear;
```

**Description**

The *Clear* method clears the work area and all the condition panels.

---

**See also:**

[DoAddRemoveLink method](#)

[DoAddRemoveTable method](#)

### 2.9.3.5 EndReload

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
procedure EndReload; virtual;
```

**Description**

The *EndReload* method should be called when the changes in the 'Criteria' panel, started by the [BeginReload](#) method are done.

---


**See also:**

[Operators property](#)

[BeginReload method](#)



## 2.9.4 Events

 Key events

[OnAddRemoveLink](#)  
[OnAddRemoveTable](#)  
[OnEnterProcParameter](#)  
[OnGetTableFields](#)  
[OnParserError](#)

### 2.9.4.1 OnAddRemoveLink

#### Applies to

[TCustomQueryBuilder](#) component

#### Declaration

##### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveLinkEvent = procedure(Sender: TObject; Link: TQBLink; Action: TQBAction; v
```

```
property OnAddRemoveLink: TAddRemoveLinkEvent;
```

#### Description

The *OnAddRemoveLink* event takes place when a link is added/removed to/from the component work area. The Link parameter defines the properties of the link to add/remove. Action defines if the link is added or removed. Output parameter Accept allows you to forbid or permit action.

---

#### See also:

[OnAddRemoveTable event](#)

#### 2.9.4.2 OnAddRemoveTable

##### Applies to

[TCustomQueryBuilder](#) component

##### Declaration

###### type

```
TQBAction = (qbaAdd, qbaRemove);
```

```
TAddRemoveTableEvent = procedure(Sender: TObject; Table: TQBTable; Action: TQBAction
```

```
property OnAddRemoveTable: TAddRemoveTableEvent;
```

##### Description

The *OnAddRemoveTable* event takes place when a table is added/removed to/from the component work area. The Table parameter defines the properties of the table to add/remove. Action defines if the table is added or removed. Output parameter Accept allows you to forbid or permit action.

---

##### See also:

[OnAddRemoveLink event](#)

### 2.9.4.3 OnEnterProcParameter

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TEnterProcParameter = procedure(Sender: TObject; SProcedure: TQBTable; ListItem: TQ
```

```
property OnEnterProcParameter: TEnterProcParameter;
```

**Description**

The *OnEnterProcParameter* event takes place before the procedure input parameter is added.

---

**See also:**

[OnGetTableFields event](#)

#### 2.9.4.4 OnGetTableFields

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration****type**

```
TGetTableFieldsEvent = procedure(const NameOfTable: string; var ADataSet: TDataSet;
```

```
property OnGetTableFields: TGetTableFieldsEvent;
```

**Description**

The *OnGetTableFields* event takes place, when you set the [SQL](#) property in run-time. Constant parameter *NameOfTable* is the table name, used in the SELECT ... FROM statement. Using parameter *ADataSet* you can define the dataset, containing data to select from, and using *AParams* you can define the dataset parameters.

---

**See also:**

[OnEnterProcParameter](#)

#### 2.9.4.5 OnParserError

**Applies to**

[TCustomQueryBuilder](#) component

**Declaration**

```
property OnParserError: TQBParserErrorEvent;
```

**Description**

The *OnParserError* event takes place when parsing the [SQL property](#) causes an error in the TQBParser object. When error occurs, you receive the following information: error code, line number, line position and token.

---

**See also:**

[Parser property](#)

[SQL property](#)

**Part**



## 3 Units

### 3.1 QBWindow unit

#### Components

[TCustomQueryBuilder](#)

[TSQLQueryBuilder](#)

[TFullQueryBuilder](#)

[TQueryBuilders](#)

[TQueryBuilder](#)

#### Objects

[TQBPalette](#)

[TQBStyle](#)

[TQBObjectStyle](#)



### 3.1.1 TQBPalette class

**Unit**


[QBWindow](#)









**Description**

The TQBPalette class contains properties defining various colors used by the descendant of the [TCustomQueryBuilder](#) component.

### 3.1.1.1 Properties

▶ Run-time only

 Key properties

-  [ActiveCondition](#)
-  [ProcClient](#)
-  [TableClient](#)
-  [Field](#)
-  [Group](#)
-  [Operation](#)
-  [Predicate](#)
-  [SubQuery](#)

## 3.1.1.1.1 ActiveCondition

**Applies to**  
[TQBPalette](#) class

**Declaration**  
`property ActiveCondition: TColor;`

**Description**

The ActiveCondition property defines the color of the active condition in the 'Criteria's' area.

---

**See also:**

[ProcClient property](#)  
[TableClient property](#)  
[Field property](#)  
[Group property](#)  
[Operation property](#)  
[Predicate property](#)  
[SubQuery property](#)

## 3.1.1.1.2 ProcClient

**Applies to**  
[TQBPalette](#) class

**Declaration**  
`property ProcClient: TColor;`

**Description**

The ProcClient property defines the procedure color the 'Builder' area.

---

**See also:**

[ActiveCondition property](#)

[TableClient property](#)

[Field property](#)

[Group property](#)

[Operation property](#)

[Predicate property](#)

[SubQuery property](#)

## 3.1.1.1.3 TableClient

**Applies to**  
[TQBPalette](#) class

**Declaration**  
`property TableClient: TColor;`

**Description**

The ProcClient property defines the table color the 'Builder' area.

---

**See also:**

[ActiveCondition property](#)

[ProcClient property](#)

[Field property](#)

[Group property](#)

[Operation property](#)

[Predicate property](#)

[SubQuery property](#)

## 3.1.1.1.4 Field

**Applies to**  
[TQBPalette](#) class

**Declaration**  
`property Field: TColor;`

**Description**

The Field property defines the color of the fields in the 'Criteria's' area.

---

**See also:**

[ActiveCondition property](#)

[ProcClient property](#)

[TableClient property](#)

[Group property](#)

[Operation property](#)

[Predicate property](#)

[SubQuery property](#)

## 3.1.1.1.5 Group

**Applies to**  
[TQBPalette](#) class

**Declaration**  
`property Group: TColor;`

**Description**

The Group property defines the color of the groups in the 'Criteria's' area.

---

**See also:**

[ActiveCondition property](#)

[ProcClient property](#)

[TableClient property](#)

[Field property](#)

[Operation property](#)

[Predicate property](#)

[SubQuery property](#)

## 3.1.1.1.6 Operation

**Applies to**  
[TQBPalette](#) class

**Declaration**  
`property Operation: TColor;`

**Description**

The Operation property defines the color of the operations in the 'Criteria's' area.

---

**See also:**

[ActiveCondition property](#)

[ProcClient property](#)

[TableClient property](#)

[Field property](#)

[Group property](#)

[Predicate property](#)

[SubQuery property](#)



## 3.1.1.1.7 Predicate

**Applies to**  
[TQBPalette](#) class

**Declaration**  
`property Predicate: TColor;`

**Description**

The Predicate property defines the color of the predicates and qualifiers in the 'Criteria' area.

---

**See also:**

[ActiveCondition property](#)

[ProcClient property](#)

[TableClient property](#)

[Field property](#)

[Group property](#)

[Operation property](#)

[SubQuery property](#)

## 3.1.1.1.8 SubQuery

**Applies to**  
[TQBPalette](#) class

**Declaration**  
`property SubQuery: TColor;`

**Description**

The SubQuery property defines the color of the subqueries in the 'Criteria's' area.

---

**See also:**

[ActiveCondition property](#)

[ProcClient property](#)

[TableClient property](#)

[Field property](#)

[Group property](#)

[Operation property](#)

[Predicate property](#)

### 3.1.2 TQBStyle class

**Unit**


[QBWindow](#)

**Description**

The TQBStyle class contains properties which define the appearance of the 'Criteria's' area buttons and 'Buider' area objects.

### 3.1.2.1 Properties

▶ Run-time only

 Key properties

-  [ButtonStyle](#)
-  [ObjectStyle](#)
-  [TableStyle](#)
-  [OldStyleLook](#)

#### 3.1.2.1.1 OldStyleLook

**Applies to**  
[TQBStyle](#) class

#### **Declaration**

```
property OldLookStyle: Boolean;
```

#### **Description**

The OldStyleLook property defines the appearance of Query Builder. It allows to hide some extended visual features of Query Builder (e.g. Subqueries tabs).

---

#### **See also:**

[ObjectStyle property](#)

[TableStyle property](#)

## 3.1.2.1.2 ButtonStyle

**Applies to**  
[TQBStyle](#) class

**Declaration****type**

```
TQBButtonStyle = (qbsUltraFlat, qbs3DLook, qbsFlat);
```

**property** ButtonStyle: TQBButtonStyle;

**Description**

The ButtonStyle property defines the appearance of the condition buttons on the 'Criteria's' panel.

---

**See also:**

[ObjectStyle property](#)

[TableStyle property](#)

## 3.1.2.1.3 ObjectStyle

**Applies to**  
[TQBStyle](#) class

**Declaration**  
`property ObjectStyle: TQBObjectStyle;`

**Description**

The ObjectStyle property defines the appearance of objects on the 'Builder' area.

---

**See also:**  
[ButtonStyle property](#)  
[TableStyle property](#)

## 3.1.2.1.4 TableStyle

**Applies to**  
[TQBStyle](#) class

**Declaration**

```
TQBTableStyle = (qtsStandard,qtsXPStyle);
```

```
property TableStyle: TQBTableStyle;
```

**Description**

The TableStyle property defines the XP style appearance of tables on the 'Builder' area.

---

**See also:**

[ButtonStyle property](#)

[ObjectStyle property](#)



### 3.1.3 TQBObjectStyle class

**Unit**


[QBWindow](#)


**Description**

The TQBObjectStyle class defines the appearance of objects on the 'Builder' area of [TCustomQueryBuilder](#) descendant.

### 3.1.3.1 Properties

▶ Run-time only

 Key properties

-  [BorderKind](#)
-  [Flat](#)
-  [FlatButtons](#)

## 3.1.3.1.1 BorderKind

**Applies to**  
[TQBObjectStyle](#) class

**Declaration****type**

```
TQBBorderKind = (qbkBump, qbkEtched, qbkRaised, qbkSunken);
```

```
property BorderKind: TQBBorderKind;
```

**Description**

The BorderKind property defines the appearance of the table and procedure borders in the 'Builder' area.

---

**See also:**

[Flat property](#)

[FlatButtons property](#)

## 3.1.3.1.2 Flat

**Applies to**  
[TQBObjectStyle](#) class

**Declaration**  
`property Flat: boolean;`

**Description**

The Flat property defines if the object on the 'Buidler' area is 'flat', i.e its borders are neither raised nor lowered.

---

**See also:**  
[BorderKind property](#)  
[FlatButtons property](#)

## 3.1.3.1.3 FlatButtons

**Applies to**  
[TQBObjectStyle](#) class

**Declaration**  
`property FlatButtons: boolean;`

**Description**

The FlatButtons property defines the appearance of the header buttons of the object on the 'Builder' area. If FlatButtons is true, then button's borders aren't raised, until you drag a mouse over it.

---

**See also:**  
[BorderKind property](#)  
[Flat property](#)

### 3.1.4 TQueryBuilder

**Unit**

[QBWindow](#)

**Description**

The TQueryBuilder is an item of [TQueryBuilders](#) collection representing a single SELECT SQL statement. TQueryBuilder stores the link to the [TCustomQueryBuilder](#) component.

### 3.1.4.1 Properties

- ▶ Run-time only
- ▶  Key properties
- ▶  [Data](#)

## 3.1.4.1.1 Data

**Applies to**  
[TQueryBuilder](#) component

**Declaration**  
`property Data: TCustomQueryBuilder;`

**Description**

The Data property is the link to TCustomQueryBuilder object representing single SELECT SQL statement.



### 3.1.5 TQueryBuilders

**Unit**

[QBWindow](#)

**Description**

The TQueryBuilders is a collection of [TQueryBuilder](#) components. Each item in this collection represents a single SELECT SQL statement.

### 3.1.5.1 Properties

▶ Run-time only                       Key properties

▶  [Items](#)

## 3.1.5.1.1 Items

**Applies to**  
[TQueryBuilder](#) component

**Declaration**  
`property Items[Index: Integer]: TQueryBuilder;`

**Description**

The QueryBuilder property is a collection of [TQueryBuilder](#) components. Each item in this collection represents a single SELECT SQL statement.

Use the Items property to access individual items in a collection. The value of the Index parameter corresponds to the Index property of a collection item. It represents its position in the collection. Items are zero-based indexed. If the Index parameter value is negative or exceeds the maximum available index, an exception is raised. The last available item index is always one less than the Count property value.


---

**See also:**

[Add method](#)

[FindByName method](#)

### 3.1.5.2 Methods

 Key methods

 [Add](#)

 [FindByName](#)

## 3.1.5.2.1 Add

**Applies to**

[TQueryBuilder](#) component

**Declaration**

```
function Add: TQueryBuilder;
```

**Description**

Adds an item to the [TQueryBuilder](#) collection.

---

**See also:**

[Items property](#)

[FindByName method](#)

## 3.1.5.2.2 FindByName

**Applies to**  
[TQueryBuilder](#) component

**Declaration**

```
function FindByName(const Name: string): TQueryBuilder;
```

**Description**

Call FindByName to retrieve field information for a field given its name. Name is the name of an existing field. FindByName returns the TQueryBuilder component that represents the specified query builder.

---

**See also:**

[Items property](#)

[Add method](#)

## 3.2 QBFloatTable unit

### Objects

[TQBTable](#)

### 3.2.1 TQBTable class

**Unit**

[QBFloatTable](#)


**Description**

The TQBTable class contains properties which describe the table object on the 'Builder' area.



### 3.2.1.1 Properties

▶ Run-time only

 Key properties

- ▶  [Alias](#)
- ▶  [Caption](#)
- ▶  [Expand](#)
- ▶  [Height](#)
- ▶  [ItemHeight](#)
- ▶  [Left](#)
- ▶  [TableName](#)
- ▶  [Top](#)
- ▶  [Width](#)

## 3.2.1.1.1 Alias

**Applies to**

[TQBTable](#) object

**Declaration**

```
property Alias: string;
```

**Description**

The Alias property is the alias of the table, which TQBTable object corresponds to.

---

**See also:**

[Caption property](#)

[TableName property](#)

## 3.2.1.1.2 Caption

**Applies to**

[TQBTable](#) object

**Declaration**

```
property Caption: string;
```

**Description**

The Caption property defines the table name, displaying on the top of the table object.

---

**See also:**

[Alias property](#)

[TableName property](#)

## 3.2.1.1.3 Expand

**Applies to**

[TQBTable](#) object

**Declaration**

```
property Expand: Bool;
```

**Description**

If Expand property is true, the object is in its usual state, otherwise, only the object header is visible.

---

**See also:**

[ItemHeight](#) property

## 3.2.1.1.4 Height

**Applies to**

[TQBTable](#) object

**Declaration**

```
property Height: Integer;
```

**Description**

The Hight property defines the object height.

---

**See also:**

[Width property](#)

[ItemHeight property](#)

## 3.2.1.1.5 ItemHeight

**Applies to**

[TQBTable](#) object

**Declaration**

```
property ItemHeight: Integer;
```

**Description**

The ItemHeight property defines the height of the table fields.

---

**See also:**

[Height property](#)

[Expand property](#)

## 3.2.1.1.6 Left

**Applies to**

[TQBTable](#) object

**Declaration**

```
property Left: Integer;
```

**Description**

The Left property is the outermost left coordinate of the object.

---

**See also:**

[Top property](#)

## 3.2.1.1.7 TableName

**Applies to**

[TQBTable](#) object

**Declaration**

```
property TableName: string;
```

**Description**

The TableName property is the name of the table, which the object corresponds to.

---

**See also:**

[Alias property](#)

[Caption property](#)



## 3.2.1.1.8 Top

**Applies to**

[TQBTable](#) object

**Declaration**

```
property Top: Integer;
```

**Description**

The Top property is the outermost top coordinate of the object.

---

**See also:**

[Left property](#)

## 3.2.1.1.9 Width

**Applies to**

[TQBTable](#) object

**Declaration**

```
property Width: Integer;
```

**Description**

The Width property defines the object width.

---

**See also:**

[Height property](#)

### 3.3 QBIBWindow unit

#### Components

[TInterBaseQueryBuilder](#)

## 3.4 QBMyWindow unit

### Components

[TMySQLQueryBuilder](#)

## 3.5 QBPgWindow unit

### Components

[TPgSQLQueryBuilder](#)

## 3.6 QBMSWindow unit

### Components

[TMSSQLQueryBuilder](#)

## 3.7 QDb2Window unit

### Components

[TDb2QueryBuilder](#)

## 3.8 QBDbiWindow unit

### Components

[TDbiQueryBuilder](#)



# Credits

**Software Developers:**

*Alexey Butalov*

*Alex Paclin*

*Dmitry Ziborov*

*Ivan Plyusnin*

*Nikolay Chanov*

*Igor Brynskich*

*Vadim Vinokur*

**Technical Writers:**

*Dmitry Doni*

*Semyon Slobodenyuk*

*Olga Ryabova*

**Cover Designer:**

*Tatyana Makurova*

**Translators:**

*Anna Shulkina*

*Sergey Fominykh*

**Team Coordinators:**

*Alexey Butalov*

*Alexander Chelyadin*

*Roman Tkachenko*