

# Navicat™

## Version 11

User Guide



# Table of Contents

<b>Getting Started</b>	<b>8</b>
System Requirements	9
Registration	9
Installation	10
Maintenance/Upgrade	11
End-User License Agreement	11
<b>Connection</b>	<b>17</b>
Navicat Cloud	18
General Settings	21
Advanced Settings	24
SSL Settings	27
SSH Settings	28
HTTP Settings	29
<b>Server Objects</b>	<b>31</b>
MySQL/MariaDB Objects	31
MySQL Tables	31
MySQL/MariaDB Table Fields	32
MySQL/MariaDB Table Indexes	34
MySQL/MariaDB Table Foreign Keys	35
MySQL/MariaDB Table Triggers	36
MySQL/MariaDB Table Options	37
MySQL/MariaDB Views	40
MySQL/MariaDB Functions/Procedures	41
MySQL/MariaDB Events	43
Oracle Objects	44
Oracle Data Pump (Available only in Full Version)	44
Oracle Data Pump Export	45
Oracle Data Pump Import	48
Oracle Debugger (Available only in Full Version)	52
Oracle Physical Attributes/Default Storage Characteristics	53
Oracle Tables	55
Oracle Normal Tables	55
Oracle Table Fields	55
Oracle Table Indexes	57
Oracle Table Foreign Keys	58
Oracle Table Uniques	59
Oracle Table Checks	59
Oracle Table Triggers	60
Oracle Table Options	61
Oracle External Tables	62

Fields for Oracle External Tables	62
External Properties for Oracle External Tables	63
Access Parameters for Oracle External Tables	64
Oracle Index Organized Tables	64
Options for Oracle Index Organized Tables	64
Oracle Views	65
Oracle Functions/Procedures	66
Oracle Database Links	68
Oracle Indexes	68
Oracle Java	71
Oracle Materialized Views	72
Oracle Materialized View Logs	75
Oracle Packages	76
Oracle Sequences	77
Oracle Synonyms	78
Oracle Triggers	78
Oracle Types	81
Oracle XML Schemas	82
Oracle Recycle Bin	83
Oracle Directories	84
Oracle Tablespaces	84
Oracle Public Database Links	87
Oracle Public Synonyms	87
PostgreSQL Objects	87
PostgreSQL Debugger (Available only in Full Version)	88
PostgreSQL Schemas	89
PostgreSQL Tables	89
PostgreSQL Normal Tables	90
PostgreSQL Table Fields	90
PostgreSQL Table Indexes	92
PostgreSQL Table Foreign Keys	93
PostgreSQL Table Uniques	94
PostgreSQL Table Checks	95
PostgreSQL Table Excludes	96
PostgreSQL Table Rules	97
PostgreSQL Table Triggers	98
PostgreSQL Table Options	99
PostgreSQL Foreign Tables	100
Fields for PostgreSQL Foreign Tables	100
Checks for PostgreSQL Foreign Tables	102
Triggers for PostgreSQL Foreign Tables	102
Table Options for PostgreSQL Foreign Tables	104

PostgreSQL Views	104
PostgreSQL Functions	106
PostgreSQL Aggregates	108
PostgreSQL Conversions	109
PostgreSQL Domains	110
PostgreSQL Indexes	111
PostgreSQL Materialized Views	113
PostgreSQL Operators	114
PostgreSQL Operator Classes	115
PostgreSQL Sequences	117
PostgreSQL Triggers	117
PostgreSQL Trigger Functions	119
PostgreSQL Types	121
PostgreSQL Tablespaces	124
PostgreSQL Casts	125
PostgreSQL Foreign Servers	125
PostgreSQL Languages	126
SQLite Objects	127
SQLite Tables	128
SQLite Table Fields	128
SQLite Table Indexes	130
SQLite Table Foreign Keys	131
SQLite Table Uniques	132
SQLite Table Checks	133
SQLite Table Triggers	134
SQLite Table Options	134
SQLite Views	135
SQLite Indexes	136
SQLite Triggers	137
SQL Preview	138
Maintain	138
Maintain MySQL/MariaDB	138
Maintain Oracle	139
Maintain PostgreSQL	143
Maintain SQLite	144
<b>Table Viewer</b>	<b>146</b>
Grid View	146
Using Navigation Bar	146
Editing Records	148
Sorting/Finding/Replacing Records	153
Filtering Records	Error! Bookmark not defined.
Manipulating Raw Data	156

Formatting Table Grid	156
Form View (Available only in Full Version)	158
Assistant Editor	158
Filter Wizard	159
<b>Query</b>	<b>161</b>
Query Builder (Available only in Full Version)	161
Query Editor	163
Editor Advanced Features	163
Query Results	167
Query Parameters	168
Debugging Oracle Query (Available only in Full Version)	168
<b>Model (Available only in Navicat Premium and Enterprise Version)</b>	<b>169</b>
Model Sidebar	169
Model Explorer Pane	169
Model History Pane	170
Model Properties Pane	170
Model Overview Pane	173
Diagram Canvas	173
Create Tables	173
Create Views	174
Create Foreign Keys	175
Create Labels	176
Create Notes	177
Create Images	178
Create Shapes	178
Create Layers	180
Format Diagram	180
Preview and Print Model	181
Reverse Engineering	182
Script Generation	182
General Settings for Export SQL	182
Advanced Settings for Export SQL	182
Forward Engineering	184
Selecting Synchronization Type	184
Selecting Schemas/Objects and Connection	184
Selecting Compare and Execution Options	184
Viewing Comparison Result	186
Model Conversion	187
Model Hints and Tips	187
<b>Advanced Tools</b>	<b>189</b>
Import Wizard	189
Setting Import File Format (Step 1)	189

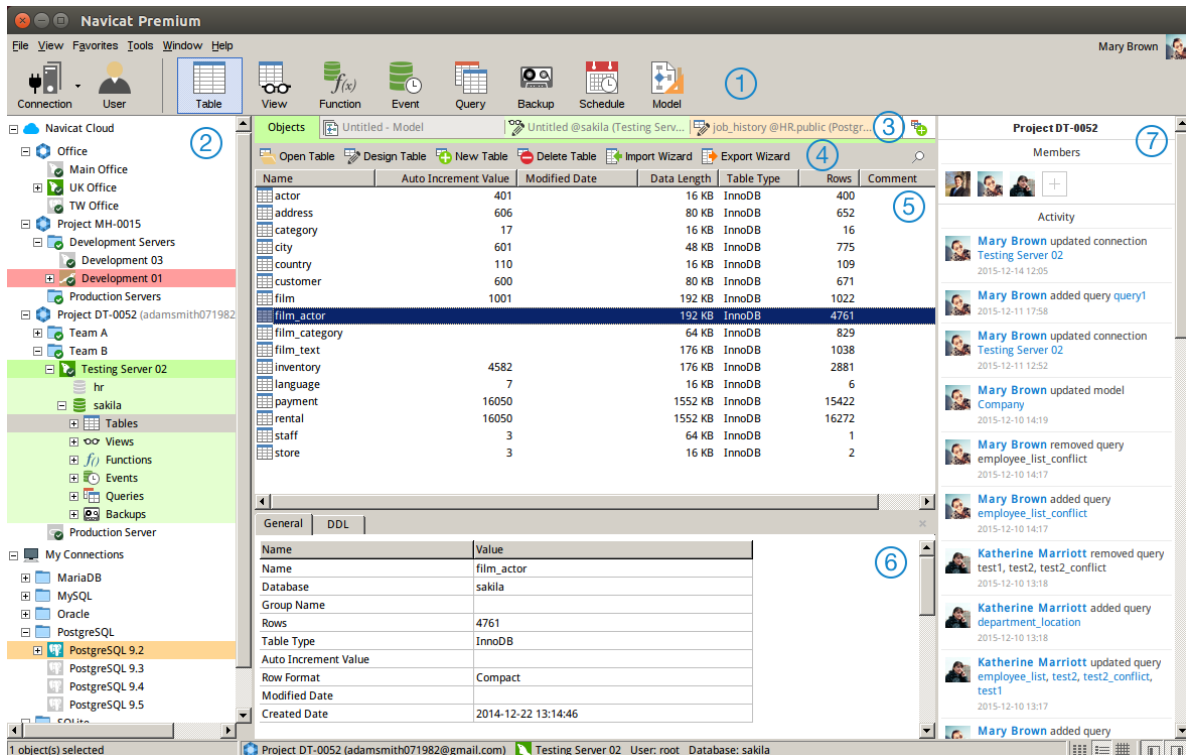
Setting Source File Name (Step 2)	189
Setting Delimiter (Step 3) - TXT, XML	189
Setting Additional Options (Step 4) - TXT, XML, HTML	190
Setting Target Table (Step 5)	191
Adjusting Field Structures and Mapping Fields (Step 6)	191
Selecting Import Mode (Step 7)	192
Saving and Confirming Import (Step 8)	193
Export Wizard	193
Setting Export File Format (Step 1)	194
Setting Destination File Name (Step 2)	194
Selecting Fields for Export (Step 3)	195
Setting Additional Options (Step 4)	195
Saving and Confirming Export (Step 5)	196
Data Transfer (Available only in Full Version)	196
General Settings for Data Transfer	196
Advanced Settings for Same Server Type Data Transfer	197
Advanced Settings for Cross Server Data Transfer (Available only in Navicat Premium)	199
Data Synchronization (Available only in Full Version)	201
General Settings for Data Synchronization	202
Advanced Settings for Data Synchronization	202
Structure Synchronization (Available only in Full Version)	202
General Settings for Structure Synchronization	202
Structure Synchronization Result	204
Backup/Restore (Available only in Full Version)	205
Backup	205
Restore	206
Extract SQL	208
Batch Job/Schedule (Available only in Full Version)	208
General Settings for Batch Job/Schedule	210
Advanced Settings for Batch Job/Schedule	210
Batch Job Converter (Available only in Navicat Premium)	211
Dump SQL File/Execute SQL File	211
Print Database/Schema/Table Structure (Available only in Full Version)	212
<b>Server Security</b>	<b>213</b>
MySQL/MariaDB Security	213
MySQL/MariaDB User Designer	213
Oracle Security	215
Oracle User Designer	215
Oracle Role Designer	217
PostgreSQL Security	218
PostgreSQL Server 7.3 to 8.0	218
PostgreSQL User Designer	218

PostgreSQL Group Designer	219
PostgreSQL Server 8.1 or above	220
PostgreSQL Role Designer	220
SQLite Security	221
SQLite User Designer	222
Privilege Manager	222
<b>Useful Tools</b>	<b>223</b>
List/Detail/ER Diagram View	223
Object Information	225
Server Monitor (Available only in Full Version)	226
Virtual Grouping (Available only in Full Version)	227
Connection Colorings	227
Favorites (Available only in Full Version)	227
Find in Database/Schema (Available only in Full Version)	228
Search Filter	228
<b>Options</b>	<b>230</b>
General Options	230
Appearance Options	231
Model Options (Available only in Full Version)	236
Miscellaneous Options	237
<b>Commands (Available only in Full Version)</b>	<b>239</b>
<b>Hot Keys</b>	<b>241</b>
<b>Log Files</b>	<b>244</b>

# Getting Started

Navicat is a multi-connections Database Administration tool allowing you to connect to MySQL, Oracle, PostgreSQL, SQLite and/or MariaDB databases, making database administration to multiple kinds of database so easy. It also can manage Amazon RDS and Amazon Redshift. Features in Navicat are sophisticated enough to provide professional developers for all their specific needs, yet easy to learn for users who are new to database server. With its well-designed Graphical User Interface(GUI), Navicat lets you quickly and easily create, organize, access and share information in a secure and easy way.

Navicat is available on three platforms - Microsoft Windows, Mac OS X and Linux. It can connect users to local/remote server, providing several utility tools such as Data Modeling, Data Transfer, Data/Structure Synchronization, Import/Export, Backup/Restore and Schedule to facilitate the process for data maintenance. For details, visit our web-site: <https://www.navicat.com>



## 1 Navicat Main Toolbar

Navicat Main Toolbar allows you to access basic objects and features, such as connections, users, tables, backup, schedule and more. To use small icons or hide the caption, simply right-click the toolbar and disable **Use Big Icons** or **Show Caption**.

## 2 Connection

Connection pane is the basic way to navigate with connections, databases and database objects. It employs tree structure which allows you to take action upon the database and their objects through their pop-up menus quickly and easily. After login [Navicat Cloud](#) feature, the Connection pane will divide into **Navicat Cloud** and **My Connections** sections. To show the opened objects only, choose **View -> Show Only Active Objects** from the main menu. To view or hide the Connection pane, choose **View -> Show Connection** from the main menu.



### ③ Tab Bar

Tab Bar allows you to switch among Object List and the tabbed windows. You can also choose to always display pop-ups on a new tab, or to always display them in a new window. If you have multiple tabs open, you can use CTRL+TAB to easily switch to other tabs. See also [Options](#).

### ④ Object List Toolbar

Object List Toolbar provides other controls that you can use to manipulate the objects.

### ⑤ Object List

Object List pane displays a list of objects, such as tables, views, queries and so on.

### ⑥ Object Information

Object Information pane shows the detailed information of the server objects and Navicat objects. To view or hide the Object Information pane, choose **View -> Show Object Information** from the main menu.

### ⑦ Navicat Cloud Activity

Navicat Cloud Activity pane shows the project members and activities. You select a project in the Connection pane or a Navicat Cloud object in the Object List pane. To view or hide the Navicat Cloud Activity pane, choose **View -> Show Navicat Cloud Activity** from the main menu.

## System Requirements

### System Requirements for Windows

- Microsoft Windows XP SP3, Vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Server 2003, Server 2008, Server 2012

### System Requirements for Mac OS X

- Mac OS X 10.7 Lion, 10.8 Mountain Lion, 10.9 Mavericks, 10.10 Yosemite, 10.11 El Capitan
- Intel CPU

### System Requirements for Linux

- Ubuntu 12.04, Ubuntu 14.04, CentOS 6.7, CentOS 7, Fedora 22, Fedora 23, Linux Mint 13, Linux Mint 17.3, openSUSE 13.2, openSUSE 42.1

## Registration

To make it economic and efficient for you to purchase our products, over 95% of customers order Navicat via our [Online Shop](#) using major Credit Cards - MasterCard, Visa, Euro card, JCB and American Express. All Online orders are processed by **PayPal** and **WorldPay**. The VeriSign Certificate for SSL transactions provided will ensure you a secured Online transactions.

If you have ordered Navicat software and would like to review your order information, or if you have questions about ordering, payments, or shipping procedures, please contact our [Navicat Sales Department](#).

After purchase you will obtain a **Registration Key** to activate your licensed Navicat by e-mail within 24 hours after we received your order. Please make sure to enter a valid e-mail address in your order. If you have not received the keys within 24 hours, it is probably that the e-mail we sent was blocked by your email spam filter. To resend your download information and keys, please submit your registered email address to our [Customer Center](#). If you get no reply from the resend form, please contact our [Navicat Sales Department](#).

Besides, if you feel uncomfortable with providing your personal information over the Internet, we accept Purchase Order and Bank/Wire Transfer. Please visit our [Offline Order](#).

## Installation

We strongly suggest that you shut down any opened applications. This will help ensure a smooth installation.

**Note:** Installing Navicat does not include the server installation. You should download and install the server manually.

For user who has been trying our unregistered version, just simply key in the **Registration Key** (16 digit) on the pop-up Registration screen.

### Installation for Download Version

1. Open or Save the **gzip** file.
2. Extract the file into anywhere you wish.
3. Open the unzipped folder.
4. Double-click **start\_navicat** to start your Navicat. Please note that it would take a while for starting-up.
5. A Registration screen will then pop up, key in the **Registration Key** (16 digit) and click **Activate** to online activate the key.

### Installation for CD Version

1. Load the Navicat CD Installation disk into the CD-ROM drive.
2. Extract the **gzip** file into anywhere you wish.
3. Open the unzipped folder.
4. Double-click **start\_navicat** to start your Navicat. Please note that it would take a while for starting-up.
5. A Registration screen will then pop up, key in the **Registration Key** (16 digit) and click **Activate** to online activate the key.

### Migrate Navicat to new computer

1. In Navicat, choose **File -> Export Connections**. The exported file (.ncx) contains all your connection settings.
2. Backup the exported file (.ncx).
3. In Navicat, choose **Help -> Registration** and click **Deactivate** to online deactivate the key.
4. Uninstall Navicat from the existing computer.

5. Re-install Navicat in the new computer.
6. Open Navicat and choose **File -> Import Connections** in the new computer.

When a new connection is being established, Navicat will create a subfolder under the [Settings Location](#). Most files are stored within this subfolder. To look for the path, right-click the connection and choose **Edit Connection -> Advanced -> Settings Location**.

Moreover, all your saved profiles are stored under [profiles](#). To look for the path, choose **Tools -> Options -> Miscellaneous -> Profiles Location**.

## Maintenance/Upgrade

### How to purchase the maintenance plan?

Navicat Software Maintenance Plan allows Navicat users to receive priority email support, receiving software upgrades and receiving bug fix releases at no additional cost during the protected period.

Subscription to the Maintenance Plan is done at the time of your software license purchase or within 90 days as of your purchase date - it cannot be added to a previously purchased product at a later date. For details, please [click here](#).

### How to upgrade your Navicat?

If you want to upgrade installed copy of Navicat to the latest release, please choose **Help -> Check For Updates** to start the Updater. It will automatically check your installed version. If there is a new version, simply follow the steps in the Updater to upgrade your Navicat. It will replace your previous Navicat and your current settings will remain unchanged.

Or, you can submit your registered email address on the [Customer Center](#) to download the latest version installer.

## End-User License Agreement

**Note:** For the License Agreement of Navicat Cloud service, please click [here](#).

IMPORTANT: THIS SOFTWARE END USER LICENSE AGREEMENT ("EULA") IS A LEGAL AGREEMENT BETWEEN YOU (EITHER AN INDIVIDUAL OR, IF PURCHASED OR OTHERWISE ACQUIRED BY OR FOR AN ENTITY, AN ENTITY) AND PREMIUMSOFT CYBERTECH LTD..READ IT CAREFULLY BEFORE COMPLETING THE INSTALLATION PROCESS AND USING THE SOFTWARE. IT PROVIDES A LICENSE TO USE THE SOFTWARE AND CONTAINS WARRANTY INFORMATION AND LIABILITY DISCLAIMERS. BY INSTALLING AND USING THE SOFTWARE, YOU ARE CONFIRMING YOUR ACCEPTANCE OF THE SOFTWARE AND AGREEING TO BECOME BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO BE BOUND BY THESE TERMS, THEN DO NOT INSTALL THE SOFTWARE AND RETURN THE SOFTWARE TO YOUR PLACE OF PURCHASE. THIS EULA SHALL APPLY ONLY TO THE SOFTWARE SUPPLIED BY PREMIUMSOFT CYBERTECH LTD. HEREWITH REGARDLESS OF WHETHER OTHER SOFTWARE IS REFERRED TO OR DESCRIBED HEREIN.

## 1. Definitions

- a. "Non-commercial Version" means a version of the Software, so identified, for use by i) the individual who is a natural person and not a corporation, company, partnership or association or other entity or organization (ii) the individual who is a student, faculty or staff member at an educational institution, and (iii) staff of a non-profit organization or charity organization only. For purposes of this definition, "educational institution" means a public or private school, college, university and other post secondary educational establishment. A non-profit organization is an organization whose primary objective is to support an issue or matter of private interest or public concern for non-commercial purposes.
- b. "Not For Resale (NFR) Version" means a version, so identified, of the Software to be used to review and evaluate the Software, only.
- c. "PremiumSoft" means PREMIUMSOFT CYBERTECH LTD. and its licensors, if any.
- d. "Software" means only the PremiumSoft software program(s) and third party software programs, in each case, supplied by PremiumSoft herewith, and corresponding documentation, associated media, printed materials, and online or electronic documentation.
- e. "Unregistered version", "Trial version" or "Demo version" means an unregistered copy of the SOFTWARE ("UNREGISTERED SOFTWARE") which may be used by the USER for evaluation purposes for a period of fourteen (14) days following the initial installation of the UNREGISTERED SOFTWARE. At the end of the trial period ("TRIAL PERIOD"), the USER must either register the SOFTWARE or remove it from his system. The UNREGISTERED SOFTWARE may be freely copied and distributed to other users for their evaluation.
- f. "Navicat Essentials" means a version of the Software, so identified, to be used for commercial purpose.

## 2. License Grants

The licenses granted in this Section 2 are subject to the terms and conditions set forth in this EULA:

- a. Subject to Section 2(b), you may install and use the Software on a single computer; OR install and store the Software on a storage device, such as a network server, used only to install the Software on your other computers over an internal network, provided you have a license for each separate computer on which the Software is installed and run. Except as otherwise provided in Section 2(b), a license for the Software may not be shared, installed or used concurrently on different computers.
- b. In addition to the single copy of the Software permitted in Section 2(a), the primary user of the computer on which the Software is installed may make a second copy of the Software and install it on either a portable computer or a computer located at his or her home for his or her exclusive use, provided that:
  - A. the second copy of the Software on the portable or home computer (i) is not used at the same time as the copy of the Software on the primary computer and (ii) is used by the primary user solely as allowed for such version or edition (such as for educational use only),
  - B. the second copy of the Software is not installed or used after the time such user is no longer the primary user of the primary computer on which the Software is installed.
- c. In the event the Software is distributed along with other PremiumSoft software products as part of a suite of products (collectively, the "Studio"), the license of the Studio is licensed as a single product and none of the products in the Studio, including the Software, may be separated for installation or use on more than one computer.

- d. You may make one copy of the Software in machine-readable form solely for backup purposes. You must reproduce on any such copy all copyright notices and any other proprietary legends on the original copy of the Software. You may not sell or transfer any copy of the Software made for backup purposes.
- e. You agree that PremiumSoft may audit your use of the Software for compliance with these terms at any time, upon reasonable notice. In the event that such audit reveals any use of the Software by you other than in full compliance with the terms of this Agreement, you shall reimburse PremiumSoft for all reasonable expenses related to such audit in addition to any other liabilities you may incur as a result of such non-compliance.
- f. Your license rights under this EULA are non-exclusive.

### 3. License Restrictions

- a. Other than as set forth in Section 2, you may not make or distribute copies of the Software, or electronically transfer the Software from one computer to another or over a network.
- b. You may not alter, merge, modify, adapt or translate the Software, or decompile, reverse engineer, disassemble, or otherwise reduce the Software to a human-perceivable form.
- c. Unless otherwise provided herein, you may not rent, lease, or sublicense the Software.
- d. Other than with respect to a Trial / Demo Version, Non-commercial Lite Version or a Not For Resale Version of the Software, you may permanently transfer all of your rights under this EULA only as part of a sale or transfer, provided you retain no copies, you transfer all of the Software (including all component parts, the media and printed materials, any upgrades, this EULA, the serial numbers, and, if applicable, all other software products provided together with the Software), and the recipient agrees to the terms of this EULA. If the Software is an upgrade, any transfer must include all prior versions of the Software from which you are upgrading. If the copy of the Software is licensed as part of the whole Studio (as defined above), the Software shall be transferred only with and as part of the sale or transfer of the whole Studio, and not separately. You may retain no copies of the Software. You may not sell or transfer any Trial / Demo Version, Non-commercial Lite Version or Not For Resale Version of the Software.
- e. Unless otherwise provided herein, you may not modify the Software or create derivative works based upon the Software.
- f. Non-commercial Versions of the Software may not be used for, or distributed to any party for, any commercial purpose.
- g. Unless otherwise provided herein, you shall not
  - A. in the aggregate, install or use more than one copy of the Trial / Demo Version and Non-commercial Lite Version of the Software,
  - B. download the Trial / Demo Version and Non-commercial Lite Version of the Software under more than one username,
  - C. alter the contents of a hard drive or computer system to enable the use of the Trial / Demo Version of the Software for an aggregate period in excess of the trial period for one license to such Trial / Demo Version,
  - D. disclose the results of software performance benchmarks obtained using the Trial / Demo Version or Non-commercial Lite Version to any third party without PremiumSoft prior written consent, or
  - E. use the Trial / Demo Version of the Software for a purpose other than the sole purpose of determining whether to purchase a license to a commercial or education version of the software; provided, however,

notwithstanding the foregoing, you are strictly prohibited from installing or using the Trial / Demo Version or Non-commercial Lite Version of the Software for any commercial training purpose.

- h. You may only use the Not for Resale Version of the Software to review and evaluate the Software.
- i. You may receive the Software in more than one medium but you shall only install or use one medium. Regardless of the number of media you receive, you may use only the medium that is appropriate for the server or computer on which the Software is to be installed.
- j. You may receive the Software in more than one platform but you shall only install or use one platform.
- k. You shall not use the Software to develop any application having the same primary function as the Software.
- l. In the event that you fail to comply with this EULA, PremiumSoft may terminate the license and you must destroy all copies of the Software (with all other rights of both parties and all other provisions of this EULA surviving any such termination).
- m. This program may include Oracle Instant Client (OCI). You agree that you shall
  - 1. not use of the Oracle Instant Client to the business operations;
  - 2. not assign, give, or transfer the Oracle Instant Client or an interest in them to another individual or entity;
    - a. make the Programs available in any manner to any third party for use in the third party's business operations; and
    - b. title to the Programs from passing to the end user or any other party;
  - 3. not reverse engineer, disassemble or decompilation the Oracle Instant Client and duplicate the Programs except for a sufficient number of copies of each Program for your licensed use and one copy of each Program media;
  - 4. discontinue use and destroy or return to all copies of the Oracle Instant Client and documentation after termination of the Agreement;
  - 5. not publish any results of benchmark tests run on the Programs;
  - 6. comply fully with all relevant export laws and regulations of the United States and other applicable export and import laws to assure that neither the Oracle Instant Client, nor any direct product thereof, are exported, directly or indirectly, in violation of applicable laws;
  - 7. allow PremiumSoft to audit your use of the Oracle Instant Client;

#### **4. Upgrades**

If this copy of the Software is an upgrade from an earlier version of the Software, it is provided to you on a license exchange basis. You agree by your installation and use of such copy of the Software to voluntarily terminate your earlier EULA and that you will not continue to use the earlier version of the Software or transfer it to another person or entity unless such transfer is pursuant to Section 3.

#### **5. Ownership**

The foregoing license gives you limited license to use the Software. PremiumSoft and its suppliers retain all rights, title and interest, including all copyright and intellectual property rights, in and to, the Software (as an independent work and as an underlying work serving as a basis for any application you may develop), and all copies thereof. All rights not specifically granted in this EULA, including Federal and International Copyrights, are reserved by PremiumSoft and its suppliers.

#### **6. LIMITED WARRANTY AND DISCLAIMER**

- a. Except with respect to Trial / Demo Version, Non-commercial Lite Version and Not For Resale Version of the Software, PremiumSoft warrants that, for a period of thirty (30) days from the date of delivery (as evidenced by a copy of your receipt): the physical media on which the Software is furnished will be free from defects in materials and workmanship under normal use. The Software is provided "as is". PremiumSoft makes no warranties, express or implied, arising from course of dealing or usage of trade, or statutory, as to any matter whatsoever.
- b. PremiumSoft provides no remedies or warranties, whether express or implied, for Trial / Demo version, Non-commercial Lite version and the Not for Resale version of the Software. Trial / Demo version, Non-commercial Lite version and the Not for Resale version of the Software are provided "as is".
- c. Except as set Forth in the foregoing limited warranty with respect to software other than Trial/ Demo version, Non-commercial Lite version and Not for Resale version, PremiumSoft and its suppliers disclaim all other warranties and representations, whether express, implied, or otherwise, including the warranties of merchantability or fitness for a particular purpose. Also, there is no warranty of non-infringement and title or quiet enjoyment. PremiumSoft does not warrant that the Software is error-free or will operate without interruption. The Software is not designed, intended or licensed for use in hazardous environments requiring fail-safe controls, including without limitation, the design, construction, maintenance or operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, and life support or weapons systems. PremiumSoft specifically disclaims any express or implied warranty of fitness for such purposes.
- d. If applicable law requires any warranties with respect to the Software, all such warranties are limited in duration to thirty (30) days from the date of delivery.
- e. No oral or written information or advice given by PremiumSoft, its dealers, distributors, agents or employees shall create a warranty or in any way increase the scope of ANY warranty PROVIDED HEREIN.

## **7. LIMITATION OF LIABILITY**

(a) Neither PremiumSoft nor its suppliers shall be liable to you or any third party for any indirect, special, incidental, punitive or consequential damages (including, but not limited to, damages for the inability to use equipment or access data, loss of business, loss of profits, business interruption or the like), arising out of the use of, or inability to use, the Software and based on any theory of liability including breach of contract, breach of warranty, tort (including negligence), product liability or otherwise, even if PremiumSoft or its representatives have been advised of the possibility of such damages.

## **8. Third Party Software**

The Software may contain third party software which requires notices and/or additional terms and conditions. By accepting this EULA, you are also accepting the additional terms and conditions of the third party software.

## **9. General**

No PremiumSoft dealer, agent or employee is authorized to make any amendment to this EULA.

This EULA contains the complete agreement between the parties with respect to the subject matter hereof, and supersedes all prior or contemporaneous agreements or understandings, whether oral or written. You agree that any varying or additional terms contained in any purchase order or other written notification or document issued by you in

relation to the Software licensed hereunder shall be of no effect. The failure or delay of PremiumSoft to exercise any of its rights under this EULA or upon any breach of this EULA shall not be deemed a waiver of those rights or of the breach.

If any provision of this EULA shall be held by a court of competent jurisdiction to be contrary to law, that provision will be enforced to the maximum extent permissible, and the remaining provisions of this EULA will remain in full force and effect.

## **10. Basis of Bargain**

The Limited Warranty and Disclaimer and Limited Liability set forth above are fundamental elements of the basis of the agreement between PremiumSoft and you. PremiumSoft would not be able to provide the Software on an economic basis without such limitations. Such Limited Warranty and Disclaimer and Limited Liability inure to the benefit of PremiumSoft's licensors.

## **11. Term**

By downloading and/or installing this SOFTWARE, the Licensor agrees to the terms of this EULA.

This license is effective until terminated. Licensor has the right to terminate your License immediately if you fail to comply with any term of this License.

"as is". Licensor makes no warranties, express or implied, arising from course of dealing or usage of trade, or statutory, as to any matter whatsoever. In particular, any and all warranties or merchantability, fitness for a particular purpose or non-infringement of third party rights are expressly excluded.

## **12. Governing Law**

This License will be governed by the laws in force in Hong Kong. You hereby consent to the non-exclusive jurisdiction and venue sitting in Hong Kong to resolve any disputes arising under this EULA.

Should you have any questions concerning the validity of this License, please contact: [licensing@navicat.com](mailto:licensing@navicat.com). If you desire to contact the Licensor for any other reason, please contact [support@navicat.com](mailto:support@navicat.com).

PremiumSoft and other trademarks contained in the Software are trademarks or registered trademarks of PremiumSoft CyberTech Ltd. in the United States and/or other countries. Third party trademarks, trade names, product names and logos may be the trademarks or registered trademarks of their respective owners. You may not remove or alter any trademark, trade names, product names, logo, copyright or other proprietary notices, legends, symbols or labels in the Software. This EULA does not authorize you to use PremiumSoft or its licensors names or any of their respective trademarks.



# Connection

To start working with your server in Navicat, you should first establish a connection or several connections using the connection window. If you are new to the server or 'Net in general' and are not quite sure how things work, you may want to look at:

- [MySQL User Manual](#)
- [Oracle Database Documentation](#)
- [PostgreSQL User Manual](#)
- [SQLite User Manual](#)
- [MariaDB Documentation](#)

To create a new connection, click  or choose **File -> New Connection**. Then, enter the necessary information in the Connection Properties window.

After you have created your connections, your databases/schemas appear in the Connection pane. If the **Show objects in Connection pane** option is checked at the [Options](#) window, all database/schema objects are also displayed in the pane. To connect to a database/schema, simply double-click it in the pane.

**Note:** Navicat authorizes you to make connection to remote servers running on different platforms, i.e. Windows, Mac, Linux and UNIX.

You can edit the connection properties by right-click the connection and choose **Edit Connection**

## Navicat Cloud

To copy or move a connection between **My Connections** and [Navicat Cloud](#), right-click the connection and choose **Copy Connection to** or **Move Connection to**.

## Flush MySQL/MariaDB Connection

**Flush** has several variant forms that clear or reload various internal caches, flush tables, or acquire locks. To execute Flush, you must have the *Reload* privilege, see [MySQL/MariaDB Security](#).

Right-click the connection and select **Flush** from the pop-up menu.

Privileges	Reload the privileges from the grant tables in the <i>mysql</i> database.
Hosts	Empty the host cache tables. You should flush the host tables if some of your hosts change IP number or if you get the error message <i>Host 'host_name' is blocked</i> . When more than <i>max_connect_errors</i> errors occur in a row for a given host while connection to MySQL server, MySQL assumes something is wrong and blocks the host from further connection requests. Flushing the host tables allow the host to attempt to connect again.
Logs	Close and reopens all log files. If you have specified the update log file or a binary log file without an extension, the extension number of the log file will be incremented by one relative to the previous

	file. If you have used an extension in the file name, MySQL will close and reopen the update log file.
Status	Reset most status variables to zero. This is something one should only use when debugging a query.
Tables	Close all open tables and forces all tables in use to be closed.

## Testing Account

Navicat provides evaluated accounts for testing purpose.

The remote MySQL server connection settings are:

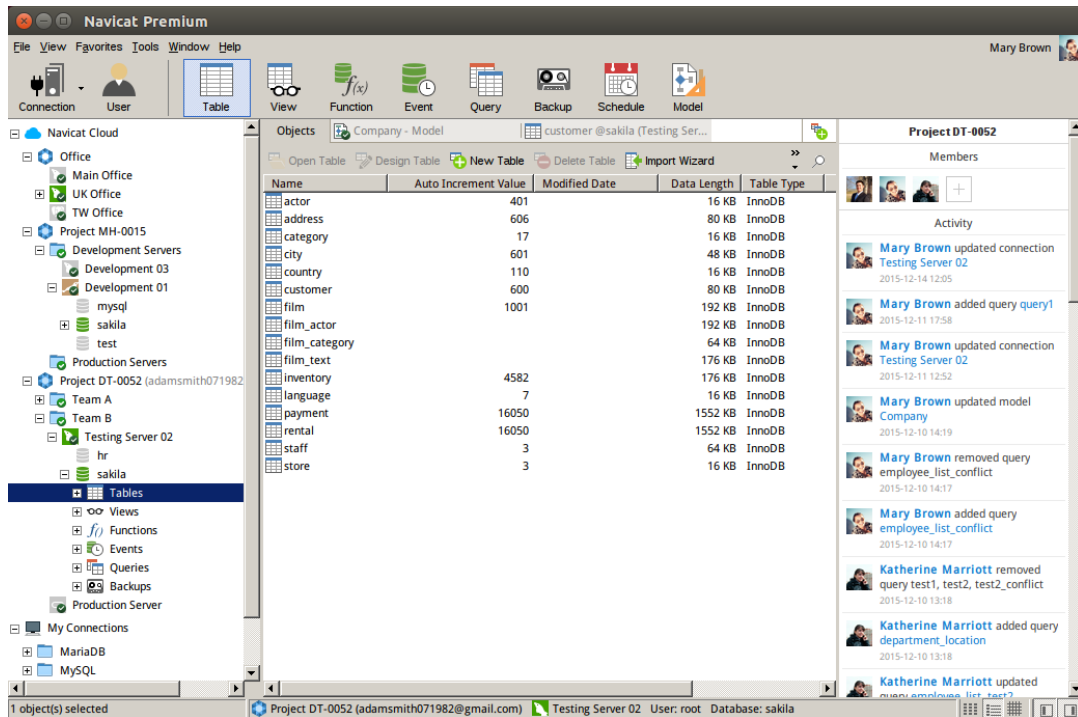
- Host Name/IP Address: server1.navicat.com
- Port: 4406
- User Name: navicat
- Password: testnavicat

The remote PostgreSQL server connection settings are:

- Host Name/IP Address: server1.navicat.com
- Port: 5432
- Initial Database: HR
- User Name: navicat
- Password: testnavicat

## Navicat Cloud

**Navicat Cloud** provides a cloud service for synchronizing Navicat connections, queries, models and virtual groups from different machines and platforms. After adding a connection to Navicat Cloud, its connection settings and queries are stored in Navicat Cloud. You can synchronize model files to Navicat Cloud and create virtual groups in Navicat Cloud. All the Navicat Cloud objects are located under different projects. You can share the project to other Navicat Cloud accounts for collaboration.



## Create a new account

1. Choose **File -> Navicat Cloud** from the main menu.
2. Click **Create Navicat ID**.
3. Enter the required information and click **Sign Up** button. A verification email will send to your email address.
4. Click the link in the email to verify the new account.

**Hint:** You can sign in with the same Navicat ID you use for the Navicat Customer Center.

## Sign in Navicat Cloud

1. Choose **File -> Navicat Cloud** from the main menu.
2. Enter your **Navicat ID** and **Password**.
3. Click **Sign In** button.
4. If you enabled two-step verification in [Navicat Cloud Portal](#) site, a code will be sent to your phone via your mobile app. Enter the received code to sign in.

## Create a project

1. Select **Navicat Cloud**.
2. Right-click it and choose **New Project**.

## Add members to a project

1. Right-click a project and choose **Collaborate with**.
2. Click **Add Members**.
3. Enter the members' Navicat ID and select the member role.
4. Click **Add**.

Member Roles	Privileges
Owner	Read Objects, Write Objects, Manage Members and Delete Project
Admin	Read Objects, Write Objects and Manage Members
Member	Read Objects and Write Objects
Guest	Read Objects

**Note:** Each time can add up to 10 members. Use comma or enter to separate the members in the edit box.

### Manage members in a project

1. Right-click a project and choose **Collaborate with**.
2. Click **Apply** after changes.

**Note:** If you are the Owner or Admin, you can click the **x** button to remove the member.

### Quit a project

1. Right-click a project and choose **Quit Project**.

### Move/Copy a connection to Navicat Cloud

1. Right-click a connection under **My Connections** and choose **Move Connection to** or **Copy Connection to**.
2. Select an existing project or create a new project.
3. The connection will move or copy to Navicat Cloud. And, all its query files will store in Navicat Cloud.

### Move/Copy a connection to My Connections

1. Right-click a connection under **Navicat Cloud** and choose **Move Connection to** or **Copy Connection to** -> **My Connections**.
2. The connection will move or copy to My Connections.

### Move a model to Navicat Cloud

1. Select a model file under **My Connections**.
2. Drag and drop the model file to a project in **Navicat Cloud**.

### Move a model to My Connections

1. Select a model file in a project under **Navicat Cloud**.
2. Drag and drop the model file to **My Connections**.

### View the cloud usage

1. Choose **File** -> **Navicat Cloud** from the main menu.

**Note:** A connection, a query, a model or a virtual group counts for one unit.

## Change your avatar

1. Choose **File** -> **Navicat Cloud** from the main menu.
2. Click the image.
3. Choose an image file.

## Manage your Navicat Cloud account

1. Choose **File** -> **Navicat Cloud** from the main menu.
2. Click your email and choose **Manage Account**.
3. A browser will open with [Navicat Cloud Portal](#) site.

## Upgrade/Manage Navicat Cloud plan

1. Choose **File** -> **Navicat Cloud** from the main menu.
2. Click **Upgrade** or **Manage Plan**.
3. A browser will open with [Navicat Cloud Portal](#) site.

## Sign out Navicat Cloud

1. Right-click **Navicat Cloud** and choose **Close All Connections** to close all connections under Navicat Cloud.
2. Choose **File** -> **Navicat Cloud** from the main menu.
3. Click **Sign Out** button.

# General Settings

To successfully establish a new connection to local/remote server - no matter via SSL, SSH or HTTP, set the connection properties in the General tab. If your Internet Service Provider (ISP) does not provide direct access to its server, Secure Tunneling Protocol (SSH) / HTTP is another solution. Enter a friendly name to best describe your connection in **Connection Name** text box.

After you logged in [Navicat Cloud](#) feature, you can choose to save the connection to **My Connections** or the projects in **Navicat Cloud** from **Add To** drop-down list. When editing a connection in Navicat Cloud, you can choose to synchronize the user name to cloud by enabling the **Sync User Name with Navicat Cloud** option.

## MySQL and MariaDB Connections

You can connect to your MySQL server remotely however for security reasons native remote direct connections to the MySQL server are disabled. Therefore, you cannot use Navicat Premium or other similar MySQL admin applications running on your computer to connect to the remote server directly unless the [User Privileges](#) has been configured.

By default, MySQL gives "root" as username and leave the password field blank.

### Host Name/IP Address

A host name where the database is situated or the IP address of the server.

**Port**

A TCP/IP port for connecting to the database server.

**User Name**

User name for connecting to the database server.

**Password**

Password for connecting to the server.

**Oracle Connection**

Navicat supports **Basic** connection type for Oracle server. In **Basic** mode, Navicat connects to Oracle through the Oracle Call Interface (OCI). OCI is an application programming interface that allows an application developer to use a third-generation language's native procedure or function calls to access the Oracle database server and control all phases of SQL statement execution. OCI is a library of standard database access and retrieval functions in the form of a dynamic-link library.

By default, Oracle created a number of user accounts upon installation. Administrative accounts: SYS, SYSTEM, SYSMAN, and DBSNMP. Sample schema accounts: SCOTT, HR, OE, OC, PM, IX and SH.

**Connection Type**

Connection type for connecting to the server: **Basic**.

**Host Name/IP Address**

A host name where the database is situated or the IP address of the server.

**Port**

A TCP/IP port for connecting to the database server.

**Service Name/SID**

Set the Service Name/SID which the user connects when making connection. Select the corresponding radio button.

**User Name**

User name for connecting to the database server.

**Password**

Password for connecting to the server.

See also:

[OCI Options](#)

**PostgreSQL Connection**

For security reasons native remote direct connections to the PostgreSQL server are disabled. Therefore, you may not be able to use Navicat Premium or other similar PostgreSQL admin applications running on your computer to connect to the remote server. By default, PostgreSQL only allows connections from the local machine using TCP/IP

connections. Other machines will not be able to connect unless you modify *listen\_addresses* in the *postgresql.conf* file, enable host-based authentication by modifying the *\$PGDATA/pg\_hba.conf* file, and restart the server. For more information: [Client Authentication](#)

By default, PostgreSQL gives "postgres" as username and leave the password field blank.

### **Host Name/IP Address**

A host name where the database is situated or the IP address of the server.

### **Port**

A TCP/IP port for connecting to the database server.

### **Initial Database**

The initial database to which user connects when making connection.

### **User Name**

User name for connecting to the database server.

### **Password**

Password for connecting to the server.

## **SQLite Connection**

You can choose the **Type** of the SQLite database and specify the file path.

### **Existing Database File**

Connect an existing database in the **Database File**.

### **New SQLite 3**

Create a new SQLite 3 database in the **Database File**.

### **New SQLite 2**

Create a new SQLite 2 database in the **Database File**.

### **Database File**

Specify the initial database file. If the HTTP Tunnel is enabled, you need to enter an absolute file path of the database file in your web server.

### **User Name**

User name for connecting to an existing database.

### **Password**

Password for connecting to an existing database.

# Advanced Settings

## Settings Location

When a new connection is being established, Navicat will create a subfolder under the Settings Location. Most files are stored within this subfolder:

Navicat Objects	Server Types	File Extensions
Query	All	.sql
Export Query Result Profile	MySQL	.npeq
	Oracle	.nopeq
	PostgreSQL	.nppeq
	SQLite	.nlpeq
	MariaDB	.nmpeq
Export View Result Profile	MySQL	.npev
	Oracle	.nopev
	PostgreSQL	.nppev
	SQLite	.nlpev
	MariaDB	.nmpev
Backup	MySQL, PostgreSQL, SQLite and MariaDB	compressed (.psc), uncompressed (.psb)
Backup Profile	MySQL	.npb
	PostgreSQL	.nppb
	SQLite	.nlpb
	MariaDB	.nmpb
Report	All	.rtm
Import Wizard Profile	MySQL	.npi
	Oracle	.nopi
	PostgreSQL	.nppi
	SQLite	.nlpi
	MariaDB	.nmpi
Export Wizard Profile	MySQL	.npe
	Oracle	.nope
	PostgreSQL	.nppe
	SQLite	.nlpe
	MariaDB	.nmpe
Export Materialized View Profile	Oracle	.nopem
	PostgreSQL	.nppem
ER Diagram File	All	.ned
Data Pump Export Profile	Oracle	.exp



Other files are located in the **profiles** directory. To look for the path, choose **Tools -> Options -> Miscellaneous -> Profiles Location**.

Other Files	Server Types	File Extensions
Data Transfer	MySQL	.npt
	Oracle	.nopt
	PostgreSQL	.nppt
	SQLite	.nlpt
	MariaDB	.nmpt
	Premium (Cross Server)	.napt
Data Synchronization	MySQL	.npd
	Oracle	.nopd
	PostgreSQL	.nppd
	SQLite	.nlpd
	MariaDB	.nmpd
Structure Synchronization	MySQL	.nps
	Oracle	.nops
	PostgreSQL	.npps
	MariaDB	.nmps
Batch Job	MySQL	.npj
	Oracle	.nopj
	PostgreSQL	.nppj
	SQLite	.nlpj
	MariaDB	.nmpj
	Premium (Cross Server)	.napj
Model File	All	.ndm
Virtual Grouping	All	vgroup.json - stores how the objects are categorized.

See also:

[Log Files](#)

### Auto Connect

With this option on, Navicat automatically open connection with the registered database at application startup.

### MySQL and MariaDB Connections

### Encoding

Choose a codepage to communicate with MySQL Server while MySQL character set not being employed.

### Keepalive Interval (sec)

This option allows you to keep the connection with the server alive by pinging it. You can set the period between pings in the edit field.

## Use Compression

This option allows you to use compression protocol. It is used if both client and server support zlib compression, and the client requests compression.

## Use Named Pipe, Socket

With this option on, Navicat uses socket file for localhost connection.

## Oracle Connection

### Role

Indicate that the database user is connecting with either the **Default**, **SYSOPER** or **SYSDBA** system privilege.

### Keepalive Interval (sec)

This option allows you to keep the connection with the server alive by pinging it. You can set the period between pings in the edit field.

### OS Authentication

With this option on, Oracle Database uses Windows user login credentials to authenticate database users.

## PostgreSQL Connection

### Keepalive Interval (sec)

This option allows you to keep the connection with the server alive by pinging it. You can set the period between pings in the edit field

## SQLite connection

### Encrypted

Enable this option and provide **Password** when connecting to an encrypted SQLite database.

### Attached Database

To attach or detach databases in the connection.

## Advanced Database Properties for MySQL, PostgreSQL and MariaDB

Set the advanced database properties, which are not obligatory. To start working with advanced database settings, check the **Use Advanced Connections**. The detailed description is given below:

To show the selected databases in the **close** state in the Connection pane, click the preferable databases in the Databases list box. The check box will show as ☒

To show the selected databases in the **open** state in the Connection pane, double-click the preferable databases in the Databases list box. The check box will show as ☒

To add a hidden database

1. Click **Add DB to List** button.
2. Enter the database name.
3. Select the newly added database in the Databases list box.

To remove a database, select the database in the Databases list box and click **Remove DB from List** button.

**Note:** The database will be just removed from the Databases list box, it will still exist in the server.

### Advanced Database Properties for SQLite

You can click **Attach Database** button to attach a database file.

Option	Description
Database File	Set the file path for a database.
Database Name	Enter the database name which displays in Navicat.
Encrypted	Enable this option and provide <b>Password</b> when connecting to an encrypted SQLite database.

To detach a database, select it from the list and click **Detach Database** button.

## SSL Settings

Secure Sockets Layer(SSL) is a protocol for transmitting private documents via the Internet. To get a secure connection, the first thing you need to do is to install OpenSSL Library and download Database Source.

**Note:** Available only for MySQL, PostgreSQL and MariaDB.

Support from PostgreSQL 8.4 or later.

### MySQL and MariaDB Connections

To provide authentication details, enable **Use Authentication** and fill in the required information:

#### Client Key

The SSL key file in PEM format to use for establishing a secure connection.

#### Client Certificate

The SSL certificate file in PEM format to use for establishing a secure connection.

#### CA Certificate

The path to a file in PEM format that contains a list of trusted SSL certificate authorities.

#### Verify CA Certificate Name

Check the server's Common Name value in the certificate that the server sends to the client.

### Specified Cipher

A list of permissible ciphers to use for SSL encryption.

### PostgreSQL Connection

Choose the **SSL Mode**:

require	Only try an SSL connection.
verify-ca	Only try an SSL connection, and verify that the server certificate is issued by a trusted CA.
verify-full	Only try an SSL connection, verify that the server certificate is issued by a trusted CA and that the server hostname matches that in the certificate.

To provide authentication details, enable **Use Authentication** and fill in the required information:

#### Client Key

The path of the client private key.

#### Client Certificate

The path of the client certificate.

#### CA Certificate

The path of the trusted certificate authorities.

#### Certificate Revocation List

The file path of the SSL certificate revocation list (CRL).

## SSH Settings

**Secure SHell (SSH)** is a program to log in into another computer over a network, execute commands on a remote server, and move files from one machine to another. It provides strong authentication and secure encrypted communications between two hosts, known as **SSH Port Forwarding (Tunneling)**, over an insecure network. Typically, it is employed as an encrypted version of Telnet.

In a Telnet session, all communications, including username and password, are transmitted in plain-text, allowing anyone to listen-in on your session and steal passwords and other information. Such sessions are also susceptible to session hijacking, where a malicious user takes over your session once you have authenticated. SSH serves to prevent such vulnerabilities and allows you to access a remote server's shell without compromising security.

**Note:** Available only for MySQL, Oracle, PostgreSQL and MariaDB.

Please make sure that the parameter - "AllowTcpForwarding" in the Linux server must be set to value "yes", otherwise, the SSH port forwarding will be disabled. To look for the path: **/etc/ssh/sshd\_config**. By default, the SSH port forwarding should be enabled. Please double check the value settings.

**\*\*** Even the server support SSH tunnel, however, if the port forwarding being disabled, Navicat cannot connect via SSH Port 22.

### Host Name/IP Address

A host where SSH server is activated.

### Port

A port where SSH server is activated, by default it is 22.

### User Name

A user on SSH server machine. (It is not a user of database server.)

### Authentication Method

Password	Provide the SSH server user <b>Password</b> .
Public Key	<b>Private Key</b> It is used together with your public key. The private key should be readable only by you.  <b>Passphrase</b> A passphrase is exactly like a password, except that it applies to the keys you are generating and not an account.

**Note:** Navicat host name at the General tab should be set relatively to the SSH server which provided by your database hosting company.

## HTTP Settings

HTTP Tunneling is a method for connecting to a server that uses the same protocol (http://) and the same port (port 80) as a web server does. It is used while your ISPs do not allow direct connections, but allows establishing HTTP connections.

**Note:** Available only for MySQL, PostgreSQL, SQLite and MariaDB.

### Uploading the Tunneling Script

To use this connection method, first thing you need to do is to upload the tunneling script to the web server where your server is located.

**Note:** **ntunnel\_mysql.php** (for both MySQL and MariaDB), **ntunnel\_pgsql.php** or **ntunnel\_sqlite.php** is available in the Navicat installation folder.

### Setting up HTTP Tunnel

The following instruction guides you through the process of configuring a HTTP connection.

1. Select the HTTP tab and enable **Use HTTP Tunnel**.
2. Enter URL of the tunneling script, e.g. *http://www.navicat.com/ntunnel\_mysql.php* .
3. If your server installed a Web Application Firewall, you can check the **Encode outgoing query with base64** option.
4. If the tunneling script is hosted in a password protected server or you have to access internet over a proxy server, you can provide the required authentication details in **Authentication** or **Proxy** tab.
5. Navicat host name at the General settings page should be set relatively to the HTTP server which provided by your database hosting company.

**Note:** HTTP Tunnel and SSH Tunnel cannot function simultaneously. The SSH Tunnel is disabled when you select the HTTP Tunnel and vice versa.

# Server Objects

Navicat provides powerful tools to manage server objects, such as databases, tables, views, functions, etc.

**Note:** Before working with the server objects in Navicat, you should establish the connection first.

## MySQL/MariaDB Objects

To start working with the server objects, you should create and open a connection. If the server is empty, you need to right-click the connection in the Connection pane and choose **New Database** to create a new database.

To edit an existing database properties, right-click the database in the Connection pane and choose **Edit Database**. Please notice that MySQL does not support renaming database through its interface at this moment. Access the directory in which databases being stored. By default, all databases store within a directory called **data** under MySQL Installation folder. For example: `/var/lib/mysql/`.

**Note:** You must stop MySQL before you can rename the database.

### Database Name

Set the name for a new database.


### Character set



Specify the default database character set.

### Collation

Specify the default database collation.

## MySQL Tables







Relational databases use tables to store data. All operations on data are done on the tables themselves or produce another table as the result. A table is a set of rows and columns, and their intersections are fields. From a general perspective, columns within a table describe the name and type of data that will be found by row for that column's fields. Rows within a table represent records composed of fields that are described from left to right by their corresponding column's name and type. Each field in a row is implicitly correlated with each other field in that row. Click  to open an object list for **Table**.

When open a table with graphical fields, right-click a table and select  **Open Table (Quick)** from the pop-up menu. Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell. If you do wish Navicat loads all your images while opening the table, use  **Open Table**.

To empty a table, right-click the selected table and choose **Empty Table** from the pop-up menu. This option is only applied when you wish to clear all the existing records without resetting the auto-increment value. To reset the auto-increment value while emptying your table, use **Truncate Table**.

## MySQL/MariaDB Table Fields

In the **Fields** tab, just simply click a field for editing. A right-click displays the pop-up menu or using field toolbar, you can create new, insert, move and drop the selected field. To search a field name, choose **Edit -> Find** or press CTRL+F.

Button	Description
 Add Field	Add a field to the table.
 Insert Field	Insert a field above an existing field. <b>Note:</b> Support from MySQL 3.22 or later.
 Delete Field	Delete the selected field.
 Primary Key	Set the field as a Primary Key.
 Move Up/  Move Down	Move the selected field up/down. <b>Note:</b> Support from MySQL 4.0.1 or later.

To add a new field with modification as one of the existing fields, you can right-click and select the **Duplicate Field** from the pop-up menu.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** drop-down list defines the type of the field data. See [MySQL Data Types/MariaDB Data Types](#) for details.

Use the **Length** edit box to define the length of the field and use **Decimals** edit box to define the number of digits after the decimal point (the scale) for Floating Point data type.

**Note:** Be careful when shortening the field length as losing data might be caused.

### Not null

Check this box to not allow the NULL values for the field.

### Virtual

Check this box to enable the virtual column settings for the field.

**Note:** Support from MariaDB 5.2 or later.

### Primary Key

A Primary Key is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.

## Field's Properties

**Note:** The following options depend on the field type you are chosen.



**Default**

Set the default value for the field.

**Comment**

Set any optional text describing the current field.

**Character set**

A character set is a set of symbols and encodings.

**Collation**

A collation is a set of rules for comparing characters in a character set.

**Note:** MySQL chooses the column character set and collation in the following manner:

- If both CHARACTER SET X and COLLATE Y were specified, then character set X and collation Y are used.
- If CHARACTER SET X was specified without COLLATE, then character set X and its default collation are used.
- Otherwise, the table character set and collation are used.

**Key Length**

The edit box will be enabled when Primary Key is set.

**Binary**

As of MySQL 4.1, values in CHAR and VARCHAR fields are sorted and compared according to the collation of the character set assigned to the field.

Before MySQL 4.1, sorting and comparison are based on the collation of the server character set; you can declare the field with the BINARY attribute to cause sorting and comparison to be based on the numeric values of the bytes in field values. BINARY does not affect how field values are stored or retrieved.

**Auto Increment**

The AUTO INCREMENT attribute can be used to generate a unique identity for new rows. To start with the AUTO INCREMENT value other than 1, you can set that value in Options tab.

**Unsigned**

UNSIGNED values can be used when you want to allow only non-negative numbers in a field and you need a bigger upper numeric range for the field.

As of MySQL 4.0.2, floating-point and fixed-point types also can be UNSIGNED. Unlike the integer types, the upper range of column values remains the same.

**Zerofill**

The default padding of spaces is replaced with zeros. For example, for a field declared as INT(5) ZEROFILL, a value of 4 is retrieved as 00004; for a field declared as FLOAT(20,10) ZEROFILL, a value of 0.1 is retrieved as 000000000.1000000015.

**Note:** If you specify ZEROFILL for a numeric type, MySQL automatically adds the UNSIGNED attribute to the field.

### On Update Current\_Timestamp

As of 4.1.2, you have more flexibility in deciding which TIMESTAMP field automatically is initialized and updated to the current timestamp.

### Values

Use this edit box to define the members of SET/ENUM.

### Virtual Type

Choose the virtual column's type. **VIRTUAL** means that the column is calculated on the fly when a command names it. **PERSISTENT** means that the value is physically stored in the table.

**Note:** Support from MariaDB 5.2 or later.

### Expression



Specify an expression or function to evaluate at insertion time.

**Note:** Support from MariaDB 5.2 or later.


## MySQL/MariaDB Table Indexes

Indexes are organized versions of specific columns in your tables. MySQL uses indexes to facilitate quick retrieval of records. With indexes, MySQL can jump directly to the records you want. Without any indexes, MySQL has to read the entire data file to find the correct record(s).

In the **Indexes** tab, just simply click an index field for editing. By using the index toolbar, you can create new, edit and delete the selected index field.

Button	Description
 Add Index	Add an index to the table.
 Delete Index	Delete the selected index.

Use the **Name** edit box to set the index name.

To include field(s) in the index, just simply double-click the **Fields** field or click  to open the editor for editing. Select the field(s) from the list. To remove the fields from the index, uncheck them in the same way. You can also use the arrow buttons to change the index field(s) order. The **Sub Part** edit box(s) is used to set index KEY LENGTH.

**Note:** Some of data types do not allow indexing by several fields. For example: BLOB

### Index Type

Define the type of the table index.

Normal	NORMAL indexes are the most basic indexes, and have no restraints such as uniqueness.
--------	---------------------------------------------------------------------------------------

Unique	UNIQUE indexes are the same as NORMAL indexes with one difference - all values of the indexed column(s) must only occur once.
Full Text	FULL TEXT indexes are used by MySQL in full-text searches.

### Index method

Specify an index type when creating an index, BTREE or HASH.

### Comment

Set any optional text describing the current index.

**Note:** Support from MySQL 5.5.3 or later.



## MySQL/MariaDB Table Foreign Keys

A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.

In the **Foreign Keys** tab, just simply click a foreign key field for editing. By using the foreign key toolbar, you can create new, edit and delete the selected foreign key field.


**Note:** Foreign Key support from MySQL 3.23.44 or later.

Editing foreign key is supported from MySQL 4.0.13 or later.

Button	Description
 Add Foreign Key	Add a foreign key to the table. <b>Note:</b> Both tables must be <i>InnoDB</i> type (or <i>solidDB</i> type if you have <a href="#">solidDB for MySQL</a> ). In the referencing table, there must be an index where the foreign key columns are listed as the first columns in the same order. Starting with MySQL 4.1.2, such an index will be created on the referencing table automatically if it does not exist.
 Delete Foreign Key	Delete the selected foreign key. <b>Note:</b> Support from MySQL 4.0.13 or later.

Use the **Name** edit box to enter a name for the new key.

Use the **Referenced Database** and **Referenced Table** drop-down lists to select a foreign database and table respectively.

To include field(s)/referenced field(s) to the key, just simply double-click the **Fields/Referenced Fields** field or click  to open the editor(s) for editing.

The **On Delete** and **On Update** drop-down list define the type of the actions to be taken.

CASCADE	Delete the corresponding foreign key, or update the corresponding foreign key to the new value of the primary key.
---------	--------------------------------------------------------------------------------------------------------------------

SET NULL	Set all the columns of the corresponding foreign key to NULL.
No ACTION	Does not change the foreign key.
RESTRICT	Produce an error indicating that the deletion or update would create a foreign key constraint violation.

Related topic:





[Foreign Keys Data Selection](#)

## MySQL/MariaDB Table Triggers

A trigger is a named database object that is associated with a table and that is activated when a particular event occurs for the table.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

**Note:** Trigger is supported from MySQL 5.0.2 or later.

Button	Description
  Add Trigger	Add a trigger to the table.
  Delete Trigger	Delete the selected trigger.

Use the **Name** edit box to set the trigger name.

Use the **Fires** drop-down list to define the trigger action time. It can be **Before** or **After** to indicate that the trigger activates before or after the statement that activated it.

### Insert

The trigger is activated whenever a new row is inserted into the table. For example, **INSERT**, **LOAD DATA**, and **REPLACE** statements.

### Update

The trigger is activated whenever a row is modified. For example, **UPDATE** statement.

### Delete

The trigger is activated whenever a row is deleted from the table. For example, **DELETE** and **REPLACE** statement. However, **DROP TABLE** and **TRUNCATE** statements on the table do not activate the trigger.

The **Definition** tab defines the statement to execute when the trigger activates. To include your statement, just simply click to write. If you want to execute multiple statements, use the **BEGIN ... END** compound statement construct.

Example:

BEGIN

```
set new.capacity = new.capacity + 100;
```

```
set new.amount = new.amount + 100;
```

END

## MySQL/MariaDB Table Options

### Engine

Define the engine of the table.

### Character set

Define the type of the character set for table.

### Collation

Choose the collation for the table.

### Auto Increment

Set/Reset the **Auto Increment** value in the edit field. The auto increment value indicates the value for next record.

### Checksum/Table Checksum *(only for MyISAM and Aria tables)*

Check this option if you want MySQL to maintain a live checksum for all rows.

**Note:** Aria table is supported from MariaDB 5.1 or later.

### Page Checksum *(only for Aria table)*

Check this option if you want index and data use page checksums for extra safety.

**Note:** Support from MariaDB 5.1 or later.

### Transactional *(only for Aria table)*

Check this option if you want crash-safe.

**Note:** Support from MariaDB 5.1 or later.

### Row Format

Defines how the rows should be stored.

### Avg. Row Length

An approximation of the average row length for your table. You need to set this only for large tables with variable-size rows.

### Max Rows

The maximum number of rows you plan to store in the table. This is not a hard limit, but rather a hint to the storage engine that the table must be able to store at least this many rows.

### Min Rows

The minimum number of rows you plan to store in the table.

## Key Block Size

This option provides a hint to the storage engine about the size in bytes to use for index key blocks. The engine is allowed to change the value if necessary. A value of 0 indicates that the default value should be used.

## Pack Keys *(take effect only with MyISAM table)*

Set this option to 1 if you want to have smaller indexes. This usually makes updates slower and reads faster. Setting the option to 0 disables all packing of keys. Setting it to **DEFAULT** tells the storage engine to pack only long *CHAR*, *VARCHAR*, *BINARY*, or *VARBINARY* columns.

## Delay Key Write *(only for MyISAM table)*

Check this option if you want to delay key updates for the table until the table is closed.

## Data Directory

To specify where the MyISAM storage engine should put a table's data file.

## Index Directory

To specify where the MyISAM storage engine should put a table's index file.

## Union *(only for MRG\_MYISAM table)*

**UNION** is used when you want to access a collection of identical *MyISAM* tables as one. This works only with *MERGE* tables. You must have *SELECT*, *UPDATE*, and *DELETE* privileges for the tables you map to a *MERGE* table.

## Insert Method *(only for MRG\_MYISAM table)*

If you want to insert data into a *MERGE* table, you must specify with *INSERT\_METHOD* the table into which the row should be inserted. *INSERT\_METHOD* is an option useful for *MERGE* tables only. Use a value of **FIRST** or **LAST** to have inserts go to the first or last table, or a value of **NO** to prevent inserts.

## Connection *(only for FEDERATED table)*

To create the local table that will be federated to the remote table. You can create the local table and specify the connection string (containing the server name, login, password) to be used to connect to the remote table using the **Connection** edit box.

The *CONNECTION* string contains the information required to connect to the remote server containing the table that will be used to physically store the data. The connection string specifies the server name, login credentials, port number and database/table information.

The format the connection string is as follows:

```
scheme://user_name[:password]@host_name[:port_num]/db_name/tbl_name
```

Sample of connection strings:

```
CONNECTION='mysql://username:password@hostname:port/database/tablename'
```

```
CONNECTION='mysql://username@hostname/database/tablename'
```

```
CONNECTION='mysql://username:password@hostname/database/tablename'
```

### Tablespace *(only for ndbcluster table)*

To specify the tablespace for the storage.

**Note:** Support from MySQL 5.1.6 or later.

### Storage *(only for ndbcluster table)*

To specify type of storage used (disk or memory), and can be one of **DISK**, **MEMORY**, or **DEFAULT**.

**Note:** Support from MySQL 5.1.6 or later.

### Stats Auto Recalc

Indicate whether to automatically recalculate persistent statistics for an InnoDB table.

**Note:** Support from MariaDB 10.0 or later.

### Stats Persistent

Indicate whether the InnoDB statistics created by ANALYZE TABLE will remain on disk or not.

**Note:** Support from MariaDB 10.0 or later.


### Partition




Set the Partition options.

**Note:** Support from MySQL 5.1 or later.

Option	Description
Partition By	Select the function that is used to determine the partition: <b>HASH</b> , <b>KEY</b> , <b>LINEAR HASH</b> , <b>LINEAR KEY</b> , <b>RANGE</b> and <b>LIST</b> .
Partitions	Set the partition number.
Subpartition By	Select the function that is used to determine the subpartition: <b>Hash</b> and <b>Key</b> .
Subpartitions	Set the subpartition number.
Partition Definition	Use <b>+</b> or <b>- Partition</b> to add or delete the partition. Use <b>+</b> or <b>- Subpartition</b> to add or delete the subpartition.
Values	For range partitioning, each partition must include a VALUES LESS THAN clause; for list partitioning, you must specify a VALUES IN clause for each partition. This is used to determine which rows are to be stored in this partition.
Engine	Select the storage engine for both partition and subpartition.
Data Directory	The directory where the data for this partition are to be stored.
Index Directory	The directory where the indexes for this partition are to be stored.
Max Rows	The maximum number of rows to be stored in the partition.
Min Rows	The minimum number of rows to be stored in the partition.
Tablespace	Designate a tablespace for the partition. Used for Falcon only.
Node Group	Set the Node Group.
Comment	Enter the comment for the partition.

## MySQL/MariaDB Views

Views (including updatable views) are implemented in MySQL Server 5.0 and available in binary releases from 5.0.1 and up. Views are useful for allowing users to access a set of relations (tables) as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table). For access control to columns, you can also use the sophisticated privilege system in MySQL Server. Click  to open an object list for **View**.

Button	Description
 Preview	Preview the result of the view.
 Explain	Show the Query Plan of the view.
 Beautify SQL	Format the codes with the Beautify SQL settings in Editor.

**Note:** You can choose to show the Result tab below the editor or in a new tab by selecting **View -> Show Result -> Below Query Editor** or **In a New Tab**.

### View Builder (Available only in Full Version)

**View Builder** allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See [Query Builder](#) for details.

### Definition

You can edit the view definition as SQL statement (SELECT statement it implements).

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

### Advanced Properties

#### Algorithm

Undefined	MySQL chooses which algorithm to use. It prefers Merge over Temptable if possible, because Merge is usually more efficient and because a view cannot be updatable if a temporary table is used.
Merge	The text of a statement that refers to the view and the view definition are merged such that parts of the view definition replace corresponding parts of the statement.
Temptable	The results from the view are retrieved into a temporary table, which then is used to execute the statement.

#### Definer

The default Definer value is the user who executes the *CREATE VIEW* statement. (This is the same as `DEFINER = CURRENT_USER`.) If a user value is given, it should be a MySQL account in 'user\_name'@'host\_name' format (the same format used in the *GRANT* statement). The user\_name and host\_name values both are required.



## Security

The SQL SECURITY characteristic determines which MySQL account to use when checking access privileges for the view when the view is executed. The legal characteristic values are **Definer** and **Invoker**. These indicate that the view must be executable by the user who defined it or invoked it, respectively. The default Security value is Definer.

### Check option


Local	Restrict the Check option only to the view being defined.
Cascaded	Cause the checks for underlying views to be evaluated as well.

## View Viewer


**View Viewer** displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See [Table Viewer](#) for details.

**Note:** Transaction is only available for updatable views.

## MySQL/MariaDB Functions/Procedures

Stored routines (procedures and functions) are supported in MySQL 5.0. A stored routine is a set of SQL statements that can be stored in the server. Once this has been done, clients do not need to keep reissuing the individual statements but can refer to the stored routine instead. Click  to open an object list for **Function**.

### Function Wizard

Click the  **New Function** from the object list toolbar. The **Function Wizard** will pop up and it allows you to create a procedure/function easily. You are allowed not to show the **Function Wizard** when create new procedure/function.

1. Select the type of the routine: **Procedure** or **Function**.
2. Define the parameter(s). Set the parameter **Mode**, **Name** and/or **Type** under corresponding columns.
3. If you create a function, select the **Return Type** from the list and enter the corresponding information: **Length**, **Decimals**, **Character set** and/or **Enum**.

**Hint:** Once uncheck the **Show wizard next time**, you can go to [Options](#) to enable it.

### Definition

Definition consists of a valid SQL procedure statement. This can be a simple statement such as *SELECT* or *INSERT*, or it can be a compound statement written using *BEGIN* and *END*. Compound statements can contain declarations, loops, and other control structure statements.

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

### Parameter

Define function/procedure parameter.

## Return Type

This text box will be enabled only for creating a function. It indicates the return type of the function.

## Type

Select the stored routines you wish to create from the drop-down list, i.e. **PROCEDURE** and **FUNCTION**.

## Advanced Properties

### Security

Specify whether the routine should be executed using the permissions of the user who creates the routine or the user who invokes it.

### Definer

The default Definer value is the user who executes the *CREATE PROCEDURE* or *CREATE FUNCTION* statement. (This is the same as `DEFINER = CURRENT_USER`.) If a user value is given, it should be a MySQL account in 'user\_name'@'host\_name' format (the same format used in the *GRANT* statement). The user\_name and host\_name values both are required.

### Data Access


Several characteristics provide information about the nature of data use by the routine.

Contains SQL	Indicate that the routine does not contain statements that read or write data. It is the default if none of these characteristics is given explicitly.
No SQL	Indicate that the routine contains no SQL statements.
Reads SQL Data	Indicate that the routine contains statements that read data, but not statements that write data.
Modifies SQL Data	Indicate that the routine contains statements that may write data.

### Deterministic


A procedure or function is considered deterministic if it always produces the same result for the same input parameters, and not deterministic otherwise.

## Result

To run the procedure/function, click  **Run** on the toolbar. If the SQL statement is correct, the statement will be executed and, if the statement is supposed to return data, the **Result** tab opens with the data returned by the procedure/function. If an error occurs while executing the procedure/function, execution stops, the appropriate error message is displayed. If the function/procedure requires input parameter, the **Input Parameters** box will pop up. Use ',' to separate the parameters.

**Hint:** Navicat supports to return 10 resultsets.

## MySQL/MariaDB Events

MySQL Event Scheduler was added in MySQL 5.1.6. MySQL Events are tasks that run according to a schedule. Therefore, we sometimes refer to them as scheduled events. When you create an event, you are creating a named database object containing one or more SQL statements to be executed at one or more regular intervals, beginning and ending at a specific date and time. Conceptually, this is similar to the idea of the Windows Task Scheduler. Click  to open an object list for **Event**.

### Definition

Definition consists of a valid SQL statement. This can be a simple statement such as *SELECT* or *INSERT*, or it can be a compound statement written using *BEGIN* and *END*. Compound statements can contain declarations, loops, and other control structure statements.

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

### Definer

Specify the user account to be used when checking access privileges at event execution time. The default DEFINER value is the user who executes the *CREATE EVENT* statement. (This is the same as DEFINER = CURRENT\_USER.) If a user value is given, it should be a MySQL account in 'user\_name'@'host\_name' format (the same format used in the GRANT statement). The user\_name and host\_name values both are required.

### STATUS

You can create an event but keep it from being active using the *DISABLE* keyword. Alternatively, you may use *ENABLE* to make explicit the default status, which is active.

### ON COMPLETION

Normally, once an event has expired, it is immediately dropped. You can override this behavior by specifying *ON COMPLETION PRESERVE*. Using *ON COMPLETION NOT PRESERVE* merely makes the default non-persistent behavior explicit.

### Schedule

#### AT

*AT timestamp* is used for a one-time event. It specifies that the event executes one time only at the date and time, given as the *timestamp*, which must include both the date and time, or must be an expression that resolves to a datetime value. Use **+INTERVAL** to create an event which occurs at some point in the future relative to the current date and time.

#### EVERY

For actions which are to be repeated at a regular interval, you can use an *EVERY* clause which followed by an *interval*. (**+INTERVAL** is not used with *EVERY*.)

## STARTS

An *EVERY* clause may also contain an optional *STARTS* clause. *STARTS* is followed by a *timestamp* value which indicates when the action should begin repeating, and may also use **+INTERVAL** interval in order to specify an amount of time "from now".

For example: ***EVERY 3 MONTH STARTS CURRENT\_TIMESTAMP + 1 WEEK*** means "every three months, beginning one week from now".

## ENDS

An *EVERY* clause may also contain an optional *ENDS* clause. The *ENDS* keyword is followed by a *timestamp* value which tells MySQL when the event should stop repeating. You may also use **+INTERVAL** interval with *ENDS*.

For example: ***EVERY 12 HOUR STARTS CURRENT\_TIMESTAMP + INTERVAL 30 MINUTE ENDS CURRENT\_TIMESTAMP + INTERVAL 4 WEEK*** is equivalent to "every twelve hours, beginning thirty minutes from now, and ending four weeks from now".

**P.S.** The *timestamp* must be in the future - you cannot schedule an event to take place in the past.


The *interval* portion consists of two parts, a quantity and a \*unit of time.

\*YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |

WEEK | SECOND | YEAR\_MONTH | DAY\_HOUR | DAY\_MINUTE |

DAY\_SECOND | HOUR\_MINUTE | HOUR\_SECOND | MINUTE\_SECOND

## Oracle Objects

To start working with the server objects, you should create and open a connection. When you create a user account, you are also implicitly creating a schema for that user. A schema is a logical container for the database objects (such as tables, views, triggers, and so on) that the user creates. The schema name is the same as the user name, and can be used to unambiguously refer to objects owned by the user. Other user schemas are showed under  **Schemas**.

**Hint:** Oracle interprets non-quoted object identifiers as uppercase. In Navicat, all objects identifier will be quoted. That is, Navicat saves exactly what you have inputted.

## Oracle Data Pump (Available only in Full Version)

Oracle Data Pump technology enables very high-speed movement of data and metadata from one database to another. It includes two utilities: Data Pump Export and Data Pump Import.

Data Pump Export is a utility for unloading data and metadata into a set of operating system files called a dump file set. The dump file set can be imported only by the Data Pump Import utility. The dump file set can be imported on the same system or it can be moved to another system and loaded there.

Data Pump Import is a utility for loading an export dump file set into a target system. The dump file set is made up of one or more disk files that contain table data, database object metadata, and control information. The files are written in

a proprietary, binary format. During an import operation, the Data Pump Import utility uses these files to locate each database object in the dump file set.

Click  to open an object list for **Data Pump**.

You can change the [Directory](#) of the dump file set by right-click anywhere in the Object List pane and select **Change Directory** from the pop-up menu.

**Note:** Support from Oracle 10g or later.

Data Pump requires SYSDBA Role and the dump file set is stored in the server.

## Oracle Data Pump Export

Before execute the Data Pump Export, click **Generate SQL** button to review the SQL. Then, you can click **Execute** button to run it.

You can save the Data Pump Export settings to a profile. Simply click the **Save** button.

**Hint:** The profiles(.exp) are saved under the [Settings Location](#).

To show the hidden tabs (advanced options), check the **Show Advanced Options** box.

### General Properties

#### Job Name

The name of the job.

#### Mode

FULL	In a full database export, the entire database is unloaded. This mode requires that you have the EXP_FULL_DATABASE role.
TABLESPACE	In tablespace mode, only the tables contained in a specified set of tablespaces are unloaded. If a table is unloaded, its dependent objects are also unloaded. Both object metadata and data are unloaded.
SCHEMAS	If you have the EXP_FULL_DATABASE role, then you can specify a list of schemas and optionally include the schema definitions themselves, as well as system privilege grants to those schemas. If you do not have the EXP_FULL_DATABASE role, you can export only your own schema.
TABLE	In table mode, only a specified set of tables, partitions, and their dependent objects are unloaded. You must have the EXP_FULL_DATABASE role to specify tables that are not in your own schema. All specified tables must reside in a single schema.

### Content

ALL	Unload both data and metadata.
DATA_ONLY	Unload only table row data; no database object definitions are unloaded.
METADATA_ONLY	Unload only database object definitions; no table row data is unloaded.

### Export Data

Select the objects to export. If you select TABLE mode, choose the schema in the **Export Schema(Table Mode)** drop-down list.

### Dump Files

Add dump files to the dump file set for the export.

### Metadata Filter

**Include** or **Exclude** a set of objects from the Export operation. Choose the **Object Type** and specify the **Name Clause**.

### Data Filter

### Query

Specify a subquery that is added to the end of the SELECT statement for the table.

### Sample

Specify a percentage for sampling the data blocks to be moved.

### Remap Data

#### Table schema

The schema containing the column to be remapped.

#### Table name

The table containing the column to be remapped.

#### Column Name

The name of the column to be remapped.

#### Package Schema

The schema of the package.

#### Package Name

The name of the package.

#### Package Function

A PL/SQL package function which is called to modify the data for the specified column.

### Encryption

## Encryption Content

Specify what to encrypt in the dump file set.

ALL	Enable encryption for all data and metadata in the export operation.
DATA_ONLY	Only data is written to the dump file set in encrypted format.
ENCRYPTED_COLUMNS_ONLY	Only encrypted columns are written to the dump file set in encrypted format.
METADATA_ONLY	Only metadata is written to the dump file set in encrypted format.
NONE	No data is written to the dump file set in encrypted format.

## Encryption Mode

Option	Description
Transparent	Allow an encrypted dump file set to be created without any intervention from a database administrator (DBA), provided the required Oracle Encryption Wallet is available.
Encryption Password	Provide a password when creating encrypted dump file sets.
Dual	Create a dump file set that can later be imported using either the Oracle Encryption Wallet or the password that was specified with the ENCRYPTION_PASSWORD parameter.

## Encryption Password

Specify a key for re-encrypting encrypted table columns, metadata, or table data so that they are not written as clear text in the dump file set.

## Confirm Password

Re-type your password.

## Advanced Properties

### Thread Number

The maximum number of worker processes that can be used for the job.

### Reuse File

A preexisting file will be overwritten.

### Enable XMLCLOBS

Check this box to enable data options for XMLCLOBS.

### Version

The version of database objects to be extracted.

COMPATIBLE	The version of the metadata corresponds to the database compatibility level and the compatibility release level for feature.
LATEST	The version of the metadata corresponds to the database version.

## Compression Type

ALL	Compress both user data and metadata.
DATA_ONLY	Compress only user data in the dump file set.
METADATA_ONLY	Compress only metadata in the dump file set.
NONE	Store the dump file set in an uncompressed format.

### Transportable

Operate on metadata for tables (and their dependent objects) within a set of selected tablespaces to perform a transportable tablespace export.

### Database Link

The name of a database link to the remote database that will be the source of data and metadata for the current job.

### Estimate

Specify that the estimate method for the size of the tables should be performed before starting the job.

### Log File Directory

Choose the log file directory.

### Log File Name

Enter the name of the log file.

### Flashback SCN

System change number (SCN) to serve as transactionally consistent point for reading user data.

### Flashback Time

Either the date and time used to determine a consistent point for reading user data or a string of the form TO\_TIMESTAMP(...).

## Oracle Data Pump Import

Before execute the Data Pump Import, click **Generate SQL** button to review the SQL. Then, you can click **Execute** button to run it.

To show the hidden tabs (advanced options), check the **Show Advanced Options** box.

### General Properties

#### Job Name

The name of the job.

#### Mode

FULL	In a full database export, the entire database is unloaded. This mode requires that you have the EXP_FULL_DATABASE role.
TABLESPACE	In tablespace mode, only the tables contained in a specified set of tablespaces are unloaded. If



	a table is unloaded, its dependent objects are also unloaded. Both object metadata and data are unloaded.
SCHEMAS	If you have the EXP_FULL_DATABASE role, then you can specify a list of schemas and optionally include the schema definitions themselves, as well as system privilege grants to those schemas. If you do not have the EXP_FULL_DATABASE role, you can export only your own schema.
TABLE	In table mode, only a specified set of tables, partitions, and their dependent objects are unloaded. You must have the EXP_FULL_DATABASE role to specify tables that are not in your own schema. All specified tables must reside in a single schema.

## Content

ALL	Unload both data and metadata.
DATA_ONLY	Unload only table row data; no database object definitions are unloaded.
METADATA_ONLY	Unload only database object definitions; no table row data is unloaded.

## Table Exists Action

Specify the action to be performed when data is loaded into a preexisting table.

SKIP	The preexisting table is left unchanged.
APPEND	New rows are added to the existing rows in the table.
TRUNCATE	Rows are removed from a preexisting table before inserting rows from the Import.
REPLACE	Preexisting tables are replaced with new definitions. Before creating the new table, the old table is dropped.

## Import Data

Select the objects to import. If you select TABLE mode, specify the schema in the **Schema** text box.

## Dump Files

Add dump files to the dump file set for the import.

## Network

### Database Link

The name of a database link to the remote database that will be the source of data and metadata for the current job.

### Estimate

Specify that the estimate method for the size of the tables should be performed before starting the job.

### Flashback SCN

System change number (SCN) to serve as transactionally consistent point for reading user data.

### Flashback Time

Either the date and time used to determine a consistent point for reading user data or a string of the form TO\_TIMESTAMP(...).

## Transportable

Operate on metadata for tables (and their dependent objects) within a set of selected tablespaces to perform a transportable tablespace export.

## DataFile Path

Specify the full file specification for a datafile in the transportable tablespace set.

## Filter

## Include/Exclude

**Include** or **Exclude** a set of objects from the Import operation. Choose the **Object Type** and specify the **Name Clause**.

## Query

Specify a subquery that is added to the end of the SELECT statement for the table. If you specify a WHERE clause in the subquery, you can restrict the rows that are selected.

## Remap Data

### Remap Data

Fields	Description
Table Schema	The schema containing the column to be remapped.
Table name	The table containing the column to be remapped.
Column Name	The name of the column to be remapped.
Package Schema	The schema of the package.
Package Name	The name of the package.
Package Function	A PL/SQL package function which is called to modify the data for the specified column.

### Remap DataFiles

Specify a remapping to be applied to objects as they are processed in the specified job. Enter the **Source DataFile** and **Target DataFile**.

## Remap Objects

### Remap Schemas

Specify a remapping to be applied to schemas as they are processed in the specified job. Enter the **Source schema** and choose the **Target schema**.

### Remap TableSpaces

Specify a remapping to be applied to tablespaces as they are processed in the specified job. Enter the **Source TableSpace** and choose the **Target TableSpace**.

### Remap Tables

Specify a remapping to be applied to tables as they are processed in the specified job. Enter the **Source Table** and choose the **Target Table**.

## Advanced Properties

### Thread Number

The maximum number of worker processes that can be used for the job.

### Reuse DataFiles

Check this box to reuse existing datafiles for tablespace creation.

### Skip Unusable Indexes

Check this box to skip loading tables that have indexes that were set to the Index Unusable state (by either the system or the user).

### Streams Configuration

Check this box to import any general Streams metadata that may be present in the export dump file.

### Data Options

A bitmask to supply special options for processing the job.

### Partition Options

Specify how partitioned tables should be handled during an import operation.

NONE	Partitioning is reproduced on the target database as it existed in the source database.
DEPARTITION	Each partition or subpartition that contains storage in the job is reproduced as a separate unpartitioned table.
MERGE	Each partitioned table is re-created in the target database as an unpartitioned table.

### Version

The version of database objects to be extracted.

COMPATIBLE	The version of the metadata corresponds to the database compatibility level and the compatibility release level for feature.
LATEST	The version of the metadata corresponds to the database version.

### Encryption Password

Specify a key for re-encrypting encrypted table columns, metadata, or table data so that they are not written as clear text in the dump file set.

### Segment Attributes

Designate the segment attribute to which the transform applies.

### Storage

Designate the storage to which the transform applies.

### OID

Designate the OID to which the transform applies.

## PCTSpace

Specify a percentage multiplier used to alter extent allocations and datafile sizes. Used to shrink large tablespaces for testing purposes.

## Log File Directory

Choose the log file directory.

## Log File Name

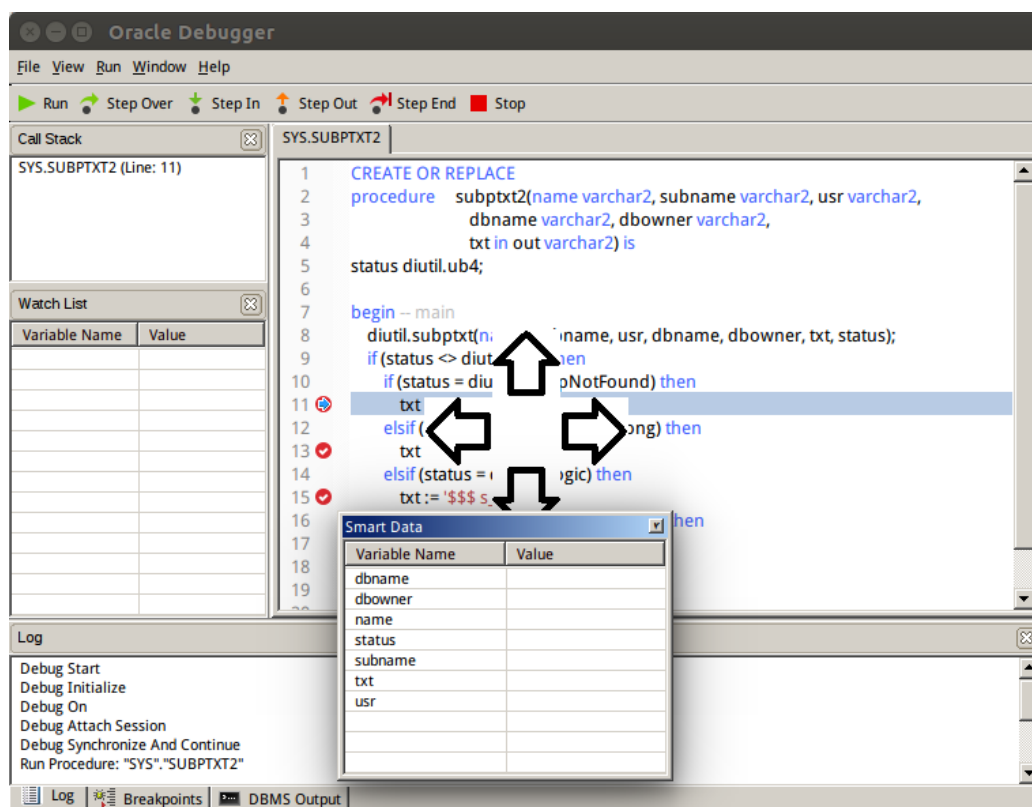
Enter the name of the log file.

## Oracle Debugger (Available only in Full Version)



Navicat provides Oracle PL/SQL debugger for debugging Oracle functions, procedures, packages and queries.





To re-arrange the workspace items, click on any pane in the workspace, then hold down your cursor and drag the pane to the desired area, and release the cursor. The selected pane will appear in its new position.


**Hint:** As you drag a pane to its new position, a **Pane Sticker** will appear and a bright blue indicator will mark the insertion point.



You can perform the most commonly used actions for debugging on the toolbar or menu:

Button	Description
 Run	Start running code in debug mode. The debugger executes your code until the end of the code or the next breakpoint is reached. Keyboard shortcut: F9
 Step Over	Resume the execution. The current line will be executed. If the line is a procedure or

	function call, it will bypass the procedure or function. The counter will then move to the next line of code. Keyboard shortcut: F8
 Step In	Resume the execution. The current line will be executed. If the line is a procedure or function call, the counter goes to the first statement in the procedure or function. Otherwise, the counter will move to the next line of code. Keyboard shortcut: F7
 Step Out	Resume the execution. The remaining part of the code within the current procedure or function will be executed. Keyboard shortcut: SHIFT+F7
 Step End	Resume the execution. The counter will jump to the last line of the procedure or function.
 Stop	Stop stepping the code. The execution will stop and cannot resume it.

The **Code** pane shows the code of the procedure/function. You can add/remove breakpoints for debugging by clicking  in the grey area beside each statement. To add a variable to the watch list, right-click the highlighted code and choose **Add to watch list**. To show the debug tips, mouse-over the code.

The **Breakpoints** pane displays all the breakpoints which allowing you to delete, enable or disable breakpoints. To enable/disable a breakpoint, check/uncheck the check box. Also, you can delete a breakpoint or all breakpoints, right-click a breakpoint and choose **Remove Breakpoint** or **Remove All Breakpoints**. To jump to the line of a breakpoint, right-click it and choose **Goto Function**.

The **Call Stack** pane displays the procedure or function calls of the current line. To jump to a procedure or function, right-click it and choose **Goto Function**.

The **DBMS Output** pane displays the results after the function or procedure has completed the execution.

The **Log** pane shows the message log when debugging the code.

The **Smart Data** pane displays information about the variables associated with breakpoints. To add a variable to the watch list, right-click a variable and choose **Add to watch list**. You can adjust the value of a watch variable by right-click the variable and choose **Adjust Value**.

The **Watch List** pane displays information about the variables being watched, allowing you to add, delete or edit watch variables. To add a watch variable, right-click anywhere of Watch List view and choose **Add Variable**. Then, enter the **Variable Name**. Also, you can right-click the highlighted code in the Code Window or the variable in the Smart Data view and choose **Add to watch list**. You can adjust the value of a watch variable by right-click the variable and choose **Adjust Value**. To delete a watch variable or all watch variables, right-click a variable and choose **Remove Variable** or **Remove All Variable**.

## Oracle Physical Attributes/Default Storage Characteristics

### % Free

Specify a whole number representing the percentage of space in each data block of the database object reserved for future updates to rows of the object.

### **% Used**

Specify a whole number representing the minimum percentage of used space that Oracle maintains for each data block of the database object. A block becomes a candidate for row insertion when its used space falls below this value.

### **Initial Transactions**

Specify the initial number of concurrent transaction entries allocated within each data block allocated to the database object.

### **Maximum Transactions**

Specify the maximum number of concurrent update transactions allowed for each data block in the segment.

### **Initial**

Specify the size of the first extent of the object. Use the drop-down list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes.

### **Next**

Specify the size of the next extent to be allocated to the object. Use the drop-down list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes.

### **Min Extents**

Specify the total number of extents to allocate when the object is created.

### **Max Extents**

Specify the total number of extents, including the first, that Oracle can allocate for the object. Check **Unlimited** if you want extents to be allocated automatically as needed.

### **Max Size**

Specify the maximum size of the storage element. Use the drop-down list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes. Check **Unlimited** if you do not want to limit the disk space of the storage element.

### **% Increase**

Specify the percent by which the third and subsequent extents grow over the preceding extent.

### **Freelists**

For objects other than tablespaces and rollback segments, specify the number of free lists for each of the free list groups for the table, partition, cluster, or index.

### **Freelist Groups**

Specify the number of groups of free lists for the database object you are creating.


### **Optimal**



Specify an optimal size for a rollback segment. Use the drop-down list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes. Check **Null** for no optimal size for the rollback segment.



## Buffer Pool

DEFAULT	Choose this to indicate the default buffer pool. This is the default for objects not assigned to KEEP or RECYCLE.
KEEP	Choose this to put blocks from the segment into the KEEP buffer pool. Maintaining an appropriately sized KEEP buffer pool lets Oracle retain the schema object in memory to avoid I/O operations. KEEP takes precedence over any NOCACHE clause you specify for a table, cluster, materialized view, or materialized view log.
RECYCLE	Choose this to put blocks from the segment into the RECYCLE pool. An appropriately sized RECYCLE pool reduces the number of objects whose default pool is the RECYCLE pool from taking up unnecessary cache space.

## Oracle Tables

Relational databases use tables to store data. All operations on data are done on the tables themselves or produce another table as the result. A table is a set of rows and columns, and their intersections are fields. From a general perspective, columns within a table describe the name and type of data that will be found by row for that column's fields. Rows within a table represent records composed of fields that are described from left to right by their corresponding column's name and type. Each field in a row is implicitly correlated with each other field in that row. Click  to open an object list for **Table**.

To create a new table, click the  **New Table** from the object list toolbar together with the  down arrow to choose the type **Normal** / **External** / **Index Organized**.

When open a table with graphical fields, right-click a table and select  **Open Table (Quick)** from the pop-up menu. Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell. If you do wish Navicat loads all your images while opening the table, use  **Open Table**.

To empty a table, right-click the selected table and choose **Empty Table** from the pop-up menu. This option is only applied when you wish to clear all the existing records without resetting the auto-increment value. To reset the auto-increment value while emptying your table, use **Truncate Table**.







## Oracle Normal Tables

Tables are the basic unit of data storage in an Oracle database. Data is stored in rows and columns. You define a table with a table name and set of columns.

In a normal (heap-organized) table, data is stored as an unordered collection (heap).

## Oracle Table Fields

In the **Fields** tab, just simply click a field for editing. A right-click displays the pop-up menu or by using the field toolbar, you can create new and drop the selected field. To search a field name, choose **Edit -> Find** or press CTRL+F.

Button	Description
 Add Field	Add a field to the table.
 Insert Field	Insert a field above an existing field. <b>Note:</b> Only available when creating a new table.
 Delete Field	Delete the selected field.
 Primary Key	Set the field as a Primary Key.
 Move Up/  Move Down	Move the selected field up/down. <b>Note:</b> Only available when creating a new table.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** drop-down list defines the type of the field data. See [Oracle Built-in Datatypes](#) for details.

Use the **Length** edit box to define the **precision** (total number of digits) of the field and use **Scale** edit box to define the **scale** (number of digits to the right of the decimal point) for **numeric** column.

**Note:** Be careful when shortening the field length as it might result in data loss.

#### Not null

Check this box to not allow the NULL values for the field.



#### Primary Key

A Primary Key is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.

#### Field's Pop-up Options

Duplicate Field	Add a new field with modification as one of the existing fields.
Primary Key Name	Enter the primary key constraint name.

#### Field's Properties

**Note:** The following options depend on the field type you are chosen.

##### Default

Set the default value for the field.

##### Comment

Set any optional text describing the current field.

##### Leading Field Precision

Set the number of digits in the leading field.



### Fractional Seconds Precision

Set the number of digits in the fractional part of the SECOND datetime field.

### Year Precision

Set the number of digits in the year.

### Unit

Set the unit either in BYTE or CHAR.

### Object Schema

Set the object schema for the field type.

### Object Type



Set the object type for the field.

## Oracle Table Indexes

Indexes are optional structures associated with tables and clusters. You can create indexes on one or more columns of a table to speed SQL statement execution on that table. An Oracle Database index provides a faster access path to table data. Indexes are the primary means of reducing disk I/O when properly used.

You can create many indexes for a table as long as the combination of columns differs for each index. You can create more than one index using the same columns if you specify distinctly different combinations of the columns.

In the **Indexes** tab, just simply click an index field for editing. By using the index toolbar, you can create new, edit and delete the selected index field.

Button	Description
 Add Index	Add an index to the table.
 Delete Index	Delete the selected index.

Use the **Name** edit box to set the index name.

To include field(s) in the index, just simply double-click the **Fields** field or click  to open the editor for editing.

### Index Type

Define the type of the table index.

Non-unique	Non-unique indexes do not impose the restriction of unique indexes on the column values.
Unique	Unique indexes guarantee that no two rows of a table have duplicate values in the key column (or columns).
Bitmap	In a bitmap index, a bitmap for each key value is used instead of a list of rowids.

## Parallel With Degree

Parallel indexing can improve index performance when you have a large amount of data, and have multiple CPUs. Enter the degree that determines the number of separate indexing processes.

## Tablespace

The tablespace in which to create the index. An index can be created in the same or different tablespace as the table it indexes.

## Schema

The schema in which to create the index.

**Note:** To create an index in your own schema, at least one of the following conditions must be true:

- The table or cluster to be indexed is in your own schema.
- You have INDEX privilege on the table to be indexed.
- You have CREATE ANY INDEX system privilege.



To create an index in another schema, all of the following conditions must be true:

- You have CREATE ANY INDEX system privilege.
- The owner of the other schema has a quota for the tablespaces to contain the index or index partitions, or UNLIMITED TABLESPACE system privilege.

## Oracle Table Foreign Keys

A foreign key specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table. We say this maintains the referential integrity between two related tables.

In the **Foreign Keys** tab, just simply click a foreign key field for editing. By using the foreign key toolbar, you can create new, edit and delete the selected foreign key field.

Button	Description
 Add Foreign Key	Add a foreign key to the table.
 Delete Foreign Key	Delete the selected foreign key.

Use the **Name** edit box to enter a name for the new key.

Use the **Referenced Schema**, **Referenced Table** and **Referenced Constraint** drop-down lists to select a foreign schema, table and constraint respectively.

To include field(s) to the key, just simply double-click the **Fields** field or click  to open the editor(s) for editing.

The **On Delete** drop-down list defines the type of the actions to be taken.

No Action	This is the default action. Referenced key values will not be updated or deleted.
-----------	-----------------------------------------------------------------------------------

CASCADE	Delete any rows referencing the deleted row, or update the value of the referencing column to the new value of the referenced column, respectively.
SET NULL	Set the referencing column(s) to null.

### Enable

You can choose whether to enable/disable the foreign key constraint by checking/unchecking the box.



Related topic:

[Foreign Keys Data Selection](#)

## Oracle Table Uniques

Unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

In the **Uniques** tab, just simply click an unique field for editing. By using the toolbar below, allowing you to create new, edit and delete the selected unique field.

Button	Description
 Add Unique	Add an unique to the table.
 Delete Unique	Delete the selected unique.

Use the **Name** edit box to set the unique name.

To set field(s) as unique, just simply double-click the **Fields** field or click  to open the editor(s) for editing.



### Enable

You can choose whether to enable/disable the unique constraint by checking/unchecking the box.

## Oracle Table Checks

A check constraint is the most generic constraint type. It allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

In the **Checks** tab, just simply click a check field for editing. By using the check toolbar, you can create new, edit and delete the selected check field.

Button	Description
 Add Check	Add a check to the table.
 Delete Check	Delete the selected check.

Use the **Name** edit box to set the check name.

## Check

Set the condition for checking, e.g. "field\_name1 > 0 AND field\_name2 > field\_name1" in the **Check** edit box. A check constraint specified as a column constraint should reference that column's value only, while an expression appearing in a table constraint may reference multiple columns.

## Definition

Type in the definition for the check constraint.



## Enable

You can choose whether to enable/disable the check constraint by checking/unchecking the box.

## Oracle Table Triggers

A trigger is a specification that the database should automatically execute a particular function whenever a certain type of operation is performed. Triggers can be defined to execute either before or after any INSERT, UPDATE, or DELETE operation, either once per modified row, or once per SQL statement.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

Button	Description
 Add Trigger	Add a trigger to the table.
 Delete Trigger	Delete the selected trigger.

## Name

Set the trigger name.

## Compound

Check to set the trigger as a compound trigger.

**Note:** Support from Oracle 11g or later.

## Row trigger

Check to set the trigger as a row trigger.

## Fires

Specify the trigger timing whether the trigger action is to be run before or after the triggering statement.

## Insert

Fire the trigger whenever an INSERT statement adds a row to a table or adds an element to a nested table.

## Update

Fire the trigger whenever an UPDATE statement changes a value in one of the columns specified in **Update Of Fields**. If no **Update Of Fields** are present, the trigger will be fired whenever an UPDATE statement changes a value in any column of the table or nested table.

### **Delete**

Fire the trigger whenever a DELETE statement removes a row from the table or removes an element from a nested table.

### **Update Of Fields**

Specify the fields for UPDATE statement trigger upon necessary.

### **Enable**

You can choose whether to enable / disable the trigger constraint by checking / unchecking the box.

### **Definition**

Type in the definition for the trigger. Example:

```
BEGIN
    add_job_history(:old.employee_id, :old.hire_date, sysdate,
        :old.job_id, :old.department_id);
END;
```

### **When Clause**

Specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger. This condition must contain correlation names and cannot contain a query.

### **Referencing Old**

Specify correlation names. The default correlation name is OLD.

### **Referencing New**

Specify correlation names. The default correlation name is NEW.

### **Follows**

Specify the relative firing order of triggers of the same type.

**Note:** Support from Orac1e 11g or later.

### **Schema**

Define the trigger on the current schema.

## **Oracle Table Options**

### **Tablespace**

Define a tablespace different from the default tablespace to create a table.

### **Logging**

Specify whether creation of a database object will be logged in the redo log file (LOGGING) or not (NOLOGGING).

## Compression

Specify whether to compress data segments to reduce disk use. It is valid only for heap-organized tables.

COMPRESS	Enable table compression.
COMPRESS FOR ALL OPERATIONS	Attempt to compress data during all DML operations on the table.
COMPRESS FOR DIRECT_LOAD OPERATIONS	Attempt to compress data during direct-path INSERT operations when it is productive to do so.
NOCOMPRESS	Disable table compression.

## Cache

Indicate how blocks are stored in the buffer cache.

CACHE	Indicate that the blocks retrieved for this table are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed.
NOCACHE	Indicate that the blocks retrieved for this table are placed at the least recently used end of the LRU list in the buffer cache when a full table scan is performed.

## Parallel With Degree

Specify the degree of parallelism, which is the number of parallel threads used in the parallel operation.

## Row Movement

With the option on, it allows the database to move a table row. It is possible for a row to move, for example, during table compression or an update operation on partitioned data.

## Physical Attributes

Refer to [Physical Attributes/Default Storage Characteristics](#).




## Oracle External Tables

External tables access data in external sources as if it were in a table in the database. While creating external tables, you are actually creating metadata in the data dictionary that enables you to access external data.

Note that external tables are read only. No DML operations are possible and no index can be created.

## Fields for Oracle External Tables

In the **Fields** tab, just simply click a field for editing. A right-click displays the pop-up menu or by using the field toolbar, you can create new and drop the selected field. To search a field name, choose **Edit -> Find** or press CTRL+F.

Button	Description
 Add Field	Add a field to the table.
 Insert Field	Insert a field above an existing field. <b>Note:</b> Only available when creating a new table.
 Delete Field	Delete the selected field.

↑ Move Up/ ↓ Move Down	Move the selected field up/down.  <b>Note:</b> Only available when creating a new table.
---------------------------	------------------------------------------------------------------------------------------------

To add a new field with modification as one of the existing fields, you can right-click and select the **Duplicate Field** from the pop-up menu.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** drop-down list defines the type of the field data. See [Oracle Built-in Datatypes](#) for details.

Use the **Length** edit box to define the **precision** (total number of digits) of the field and use **Scale** edit box to define the **scale** (number of digits to the right of the decimal point) for **numeric** column.

**Note:** Be careful when shortening the field length as it might result in data loss.

## Field's Properties

**Note:** The following options depend on the field type you are chosen.

### Leading Field Precision

Set the number of digits in the leading field.

### Fractional Seconds Precision

Set the number of digits in the fractional part of the SECOND datetime field.

### Year Precision

Set the number of digits in the year.

### Unit

Set the unit either in BYTE or CHAR.

### Object Schema

Set the object schema for the field type.

### Object Type

Set the object type for the field.

## External Properties for Oracle External Tables

### Default Directory

Specify the default directory for the external table.

### Directory

Set the external directory.

**Location**

Set the external source location.

**Access Driver**

Specify the access driver for the external table. The default type for external tables is ORACLE\_LOADER.

**Reject Limit**

Specify the limit on the number of errors that can occur during a query of the external data.

**Parallel With Degree**

Check to enable parallel query on the data sources and specify the degree of parallel access.

## Access Parameters for Oracle External Tables

Describe the mapping of the external data to the Oracle Database data columns.

**Using CLOB**

Check this option to get a CLOB data value of the returned query.

## Oracle Index Organized Tables

An index-organized table has a storage organization that is a variant of a primary B-tree. Data for an index-organized table is stored in a B-tree index structure in a primary key sorted manner. Each leaf block in the index structure stores both the key and nonkey columns.

Index-organized tables have full table functionality. They support features such as constraints, triggers etc with additional features such as key compression.

**Note:** The Table Designer for **Index Organized Tables** differs from **Normal Tables** only on the **Options** tab. Therefore, you can refer to the [Normal Table](#) on the similar tabs.

## Options for Oracle Index Organized Tables

**Tablespace**

Define a tablespace different from the default tablespace to create a table.

**Logging**

Specify whether creation of a database object will be logged in the redo log file (LOGGING) or not (NOLOGGING).

**Parallel With Degree**

Specify the degree of parallelism, which is the number of parallel threads used in the parallel operation.



## Row Movement

With the option on, it allows the database to move a table row. It is possible for a row to move, for example, during table compression or an update operation on partitioned data.

## Physical Attributes

Refer to [Physical Attributes/Default Storage Characteristics](#).

## Key Compress

Check this option to enable key compression. Upon necessary, you can also specify the prefix length (as the number of key columns), which identifies how the key columns are broken into a prefix and suffix entry.

## Mapping Table

Specify if there is a mapping table for the index-organized table. Note that a mapping table is required for creating bitmap indexes on an index-organized table.

## % Threshold

When an overflow segment is being used, it defines the maximum size of the portion of the row that is stored in the index block, as a percentage of block size.

## Overflow Properties

Check to enable an overflow storage area.

**Note:** After saving the table, this option cannot be unchecked.

## Overflow Column

Specify the column to be put in a separate overflow data segment.

## Overflow Tablespace

Specify the tablespace in which the overflow segment to be stored.


## Overflow Logging



Specify whether creation of a database object will be logged in the redo log file (LOGGING) or not (NOLOGGING).


## Overflow Physical Attributes

Refer to [Physical Attributes/Default Storage Characteristics](#).

## Oracle Views

Views are useful for allowing users to access a set of relations (tables) as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table). Click  to open an object list for **View**.

Button	Description
 Preview	Preview the result of the view.
 Explain	Show the Query Plan of the view.

 Beautify SQL	Format the codes with the Beautify SQL settings in Editor.
------------------------------------------------------------------------------------------------	------------------------------------------------------------

**Note:** You can choose to show the Result tab below the editor or in a new tab by selecting **View -> Show Result -> Below Query Editor** or **In a New Tab**.

### View Builder (Available only in Full Version)

**View Builder** allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See [Query Builder](#) for details.

### Definition

You can edit the view definition as SQL statement (SELECT statement it implements).

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

### Advanced Properties

#### Restrict Query

Specify the name. If you omit this identifier, then Oracle automatically assigns a name of the form SYS\_Cn, where n is an integer that makes the constraint name unique within the database.

#### Read Only

Indicate that the table or view cannot be updated.

#### Check option

Indicate that Oracle Database prohibits any changes to the table or view that would produce rows that are not included in the subquery.


#### Force On Create

Check this option if you want to create the view regardless of whether the base tables of the view or the referenced object types exist or the owner of the schema containing the view has privileges on them.


### View Viewer

**View Viewer** displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See [Table Viewer](#) for details.

## Oracle Functions/Procedures

A procedure or function is a schema object that consists of a set of SQL statements and other PL/SQL constructs, grouped together, stored in the database, and run as a unit to solve a specific problem or perform a set of related tasks. Procedures and functions are identical except that functions always return a single value to the caller, while procedures do not. Click  to open an object list for **Function**.

## Function Wizard

Click the  **New Function** from the object list toolbar. The **Function Wizard** will pop up and it allows you to create a procedure/function easily. You are allowed not to show the **Function Wizard** when create new procedure/function.







1. Specify the **Name** of the routine and select the type of the routine: **Procedure** or **Function**.
2. Define the parameter(s) of the procedure/function. Set the parameter **Name**, **Type**, **Mode** and **Default Value** under corresponding columns.
3. If you create a function, select the **Return Type** from the list.

**Hint:** Once uncheck the **Show wizard next time**, you can go to [Options](#) to enable it.

## Definition


The **Code Outline** window displays information about the function/procedure including parameter, code body, etc. To show the **Code Outline** window, simply choose **View -> Code Outline**.

**Note:** Available only in Full Version.

Button	Description
	Refresh the code outline.
	Show the detail view of the code outline.
	Turn mouse over highlight on or off.
	Expand the selected item.
	Collapse the selected item.
	Toggle sorting by position.


**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).


## Result

To run the procedure/function, click  **Run** on the toolbar. If the SQL statement is correct, the statement will be executed and, if the statement is supposed to return data, the **DBMS Output** tab opens with the data returned by the procedure/function. If an error occurs while executing the procedure/function, execution stops, the appropriate error message is displayed. If the function/procedure requires input parameter, the **Input Parameters** box will pop up.


**Hint:** Navicat supports to return 10 resultsets.

## Debug (Available only in Full Version)

To debug the function/procedure, click  **Debug** on the toolbar to launch the [Oracle Debugger](#). Enter the Input Parameters if necessary.

You can add/remove breakpoints for debugging by clicking  in the grey area beside each statement.

## Oracle Database Links

Database link is a named schema object that describes a path from one database to another and are implicitly used when a reference is made to a global object name in a distributed database. After you have created a database link, you can use it to refer to tables and views on the other database. Click  -> **Database Link** to open an object list for **Database Link**.

### General Properties

#### Service Name

Specify the service name of a remote database.

#### User Name

The user name used to connect to the remote database using a fixed user database link.

#### Password

The password for connecting to the remote database.


#### Current user

With this option checked, a current user database link is created. The current user must be a global user with a valid account on the remote database.

#### Shared

Fill in **Authentication username** and **Authentication password** when Shared option is enabled.

## Oracle Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table. Click  -> **Index** to open an object list for **Index**.

You can choose the index **Type**:

Normal	A normal index does not impose restrictions on the column values.
Unique	A unique index indicates that no two rows of a table have duplicate values in the key columns.
Bitmap	A bitmap index created with a bitmap for each distinct key, rather than indexing each row separately. Bitmap indexes store the rowids associated with a key value as a bitmap. Each bit in the bitmap corresponds to a possible rowid.
Domain	A domain index is an index designed for a specialized domain, such as spatial or image processing. Users can build a domain index of a given type after the designer creates the indextype.
Cluster	A cluster index is an index designed for a cluster.

### General Properties for Normal and Unique Indexes

**Table schema**

The schema that contains the index.

**Table name**

The table name.

**Columns**

Use the **Name** drop-down list to select the field name and **Order** drop-down list to define the order of the index (ASC or DESC).

**General Properties for Bitmap Index****Table schema**

The schema that contains the index.

**Table name**

The table name.

**Bitmap Join Index**

In addition to a bitmap index on a single table, you can create a bitmap join index, which is a bitmap index for the join of two or more tables. A bitmap join index is a space efficient way of reducing the volume of data that must be joined by performing restrictions in advance.

**Columns**

Use the **Schema**, **Table** and/or **Name** drop-down lists to select the schema, table and/or field name and **Order** drop-down list to define the order of the index (ASC or DESC).

**Bitmap Join**

Use the **InnerSchema**, **InnerTable**, **InnerField**, **OuterSchema**, **OuterTable** and **OuterField** drop-down lists to select joined schemas, tables and fields respectively.

**General Properties for Domain Index****Table schema**

The schema that contains the index.

**Table name**

The table name.

**Column**

The column which the index is based.

**Schema**

The schema of the indextype.

## Type

Select the created or built-in indextypes.

## Parameters

Information about the path table and about the secondary indexes corresponding to the components of XMLIndex.

## General Properties for Cluster Index

### Table schema

The schema that contains the index.

### Cluster name

The name of the cluster.

## Advanced Properties

### Unusable

An unusable index must be rebuilt, or dropped and re-created, before it can be used.

### Tablespace

The name of the tablespace to hold the index.

### Compress

To enable key compression, which eliminates repeated occurrence of key column values and may substantially reduce storage.

**Note:** No compression for Bitmap Indexes.

### Parallel

The creation of the index will be parallelized.

### Reverse

To store the bytes of the index block in reverse order, excluding the rowid.

### Logging

Choose **Logging** to log the creation of the index in the redo log file. Or, choose **No Logging** for no log.

### Visibility

Specify the index is **Visible** or **Invisible** to the optimizer.

### Online

To indicate that DML operations on the table will be allowed during creation of the index.


### No Sort

To indicate to the database that the rows are already stored in the database in ascending order, so that Oracle Database does not have to sort the rows when creating the index.

## Physical Attributes

Set the [physical attributes](#) of an index.

## Oracle Java

Java is an object-oriented programming language efficient for application-level programs. You can write and load applications within the database. Click  -> **Java** to open an object list for **Java**.

You can choose the **Type**: Java Source, Java Class or Java Resource.

### General Properties for Java Source

#### BFile

Select the **Directory** and type the **Server Filename**.

#### Load from file

Browse the **File Path** of Java source file.

#### Plain source

Type the source code in the **Source** box.

### General Properties for Java Class and Java Resource

#### BFile

Select the **Directory** and type the **Server Filename**.

#### Load from file

Browse the **File Path** of Java class or Java resource file.

### Advanced Properties

#### Invoker Rights

Select **CURRENT\_USER** to indicate that the methods of the class execute with the privileges of CURRENT\_USER or **DEFINER** to indicate that the methods of the class execute with the privileges of the owner of the schema in which the class resides, and that external names resolve in the schema where the class resides.

#### Resolver

Specify a mapping of the fully qualified Java name to a Java schema object.

#### Compile or Resolve


Check this to specify that Oracle Database should attempt to resolve the Java schema object that is created if this statement succeeds.




#### No Force

Check this to roll back the results of the CREATE command of Java if you have enabled Compile or Resolve and the

resolution or compilation fails. If you do not specify this option, then Oracle Database takes no action if the resolution or compilation fails, and the created schema object remains.

## Oracle Materialized Views

Materialized view is a schema object that can be used to summarize, compute, replicate, and distribute data. Click  -> **Materialized View** to open an object list for **Materialized View**.

Button	Description
 Preview	Preview the result of the materialized view.
 Explain	Show the Query Plan of the materialized view.
 Beautify SQL	Format the codes with the Beautify SQL settings in Editor.

**Note:** You can choose to show the Result tab below the editor or in a new tab by selecting **View -> Show Result -> Below Query Editor** or **In a New Tab**.

### View Builder (Available only in Full Version)

**View Builder** allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See [Query Builder](#) for details.

### Definition

You can edit the view definition as SQL statement (SELECT statement it implements).

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

### Advanced Properties

#### When

Demand	The materialized view will be refreshed on demand by calling one of the three DBMS_MVIEW refresh procedures.
Commit	A fast refresh is to occur whenever the database commits a transaction that operates on a master table of the materialized view.
Automatic	The database automatically refresh the materialized view with the automatic refresh time.
Never	The materialized view will not be refreshed with any Oracle Database refresh mechanism or packaged procedure.

#### Method

FORCE	When a refresh occurs, Oracle Database will perform a fast refresh if one is possible or a complete refresh if fast refresh is not possible.
FAST	A incremental refresh method, which performs the refresh according to the changes that have occurred to the master tables.



COMPLETE	A complete refresh method, which is implemented by executing the defining query of the materialized view.
----------	-----------------------------------------------------------------------------------------------------------

### Start On

A datetime expression for the first automatic refresh time.

### Next

A datetime expression for calculating the interval between automatic refreshes.

**Note:** To edit the datetime, just simply click ... and choose / enter the desired data.

### Type

Primary Key	A primary key materialized view.
Row ID	A rowid materialized view.

### Master

The remote rollback segment is used at the remote master site for the individual materialized view.

### Local

The remote rollback segment is used for the local refresh group that contains the materialized view.

### Constraints

Enforced	Oracle Database use enforced constraints during the refresh operation.
Trusted	Oracle Database use dimension and constraint information that has been declared trustworthy by the database administrator but that has not been validated by the database.

### No Index

Check this to suppress the creation of the default index.

### Build Type

Immediate	The materialized view is to be populated immediately.
Deferred	The materialized view is to be populated by the next refresh operation.
Prebuilt	To register an existing table as a preinitialized materialized view.

### Reduced Precision

With	To authorize the loss of precision that will result if the precision of the table or materialized view columns do not exactly match the precision returned by subquery.
Without	To require that the precision of the table or materialized view columns match exactly the precision returned by subquery, or the create operation will fail.

### Compress

Data segments are compressed to reduce disk and memory use.

## Parallel

Choose **NOPARALLEL** for serial execution or **PARALLEL** if you want Oracle to select a degree of parallelism equal to the number of CPUs available on all participating instances times the value of the `PARALLEL_THREADS_PER_CPU` initialization parameter.

## With Degree

Set the default degree of parallelism for queries and DML on the materialized view after creation.

## Logging

Choose **LOGGING** for logging the creation of Materialized view in the redo log file. Choose **NOLOGGING** for no logging.

## Tablespace

Choose the tablespace in which the materialized view is to be created.

## Physical Attributes

Set the [Physical Attributes](#) of the materialized view.

## Enable Cache

The blocks retrieved for the table are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed.

## For Update

Check this to allow a subquery, primary key, object, or rowid materialized view to be updated. When used in conjunction with Advanced Replication, these updates will be propagated to the master.

## Enable Query Rewrite

The materialized view is used for query rewrite.


## Using Index Clause

Option	Description
Tablespace	Choose the tablespace of the index.
Physical Attributes	Set the <a href="#">Physical Attributes</a> for the default index Oracle Database uses to maintain the materialized view data.

## View Viewer

**View Viewer** displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See [Table Viewer](#) for details.

## Oracle Materialized View Logs

Materialized view log is a schema object that records changes to a master table's data so that a [Materialized View](#) defined on the master table can be refreshed incrementally. Click  -> **Materialized View Log** to open an object list for **Materialized View Log**.

### General Properties

#### Table

The table of the materialized view log.

#### Tablespace

The tablespace of the materialized view log.

#### Logging

To specify either **LOGGING** or **NOLOGGING** to establish the logging characteristics for the materialized view log.

#### Cache

CACHE	The blocks retrieved for this log are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed.
NOCACHE	The blocks are placed at the least recently used end of the LRU list.

#### New Values

INCLUDING	To save both new and old values in the log.
EXCLUDING	To disable the recording of new values in the log.

#### Parallel With Degree

To determine the number of parallel threads used in the parallel operation.

#### Physical Attributes

Set the [Physical Attributes](#) of a materialized view log.

#### Object ID

The system-generated or user-defined object identifier of every modified row should be recorded in the materialized view log.

#### Primary Key

The primary key of all rows changed should be recorded in the materialized view log.

#### Row ID

The rowid of all rows changed should be recorded in the materialized view log.


## Sequence

A sequence value providing additional ordering information should be recorded in the materialized view log.

## Fields







Choose the fields whose values you want to be recorded in the materialized view log for all rows that are changed.

## Oracle Packages



Packages are encapsulated collections of related procedures, stored functions, and other program objects stored together in the database. Package bodies, specified subsequently, defines these objects. An package consists of two parts: a specification and a body. Click  -> **Package** to open an object list for **Package**.

The **Code Outline** window displays information about the package/package body including function, procedure, parameter, code body, etc. To show the **Code Outline** window, simply choose **View** -> **Code Outline**.

**Note:** Available only in Full Version.

Button	Description
	Refresh the code outline.
	Show the detail view of the code outline.
	Turn mouse over highlight on or off.
	Expand the selected item.
	Collapse the selected item.
	Toggle sorting by position.

## Package's Definition

Enter the package's definition. After saving the package, you can edit the Package Body. Just click  **New Package Body** or  **Design Package Body** to open the Package Body Designer.


**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

## Package Body's Definition


Enter the package body's definition. To edit the Package Specification, click  **Design Package Specification** to open the Package Designer.

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

## Result


To run the package click  **Run** on the toolbar. If the SQL statement is correct, the statement will be executed and, if the statement is supposed to return data, the **DBMS Output** tab opens with data returned by the package. If an error occurs while executing the package, execution stops, the appropriate error message is displayed. Select the function/procedure and enter the parameter(s) if the function/procedure has input parameter(s).

## Debug (Available only in Full Version)

To debug the package click  **Debug** on the toolbar to launch the [Oracle Debugger](#). Select the function/procedure and enter the parameter(s) if the function/procedure has input parameter(s).

You can add/remove breakpoints for debugging by clicking  in the grey area beside each statement.

## Oracle Sequences

Sequence involves creating and initializing a new special single-row table. It is usually used to generate unique identifiers for rows of a table. Click  -> **Sequence** to open an object list for **Sequence**.

### General Properties

#### Increment

To specify which value is added to the current sequence value to create a new value. A positive value will make an ascending sequence, a negative one a descending sequence. The default value is 1.

#### Start with

To specify the first sequence number to be generated.

#### Minimum

The minimum value a sequence can generate.

#### Maximum

The maximum value for the sequence.

#### Cache

To specify how many values of the sequence the database preallocates and keeps in memory for faster access. The minimum value for this parameter is 2.

#### No Cache

This option indicates that values of the sequence are not preallocated.


#### Cycled

This option allows the sequence continues to generate values after reaching either its maximum or minimum value. After an ascending sequence reaches its maximum value, it generates its minimum value. After a descending sequence reaches its minimum, it generates its maximum value.

#### Order

This option guarantees that sequence numbers are generated in order of request.

## Oracle Synonyms

Synonym is an alias for any table, view, materialized view, synonym, procedure, function, package, type, Java class schema object, user-defined object type, or another synonym. Because a synonym is simply an alias, it requires no storage other than its definition in the data dictionary. Click  -> **Synonym** to open an object list for **Synonym**.

### General Properties

#### Object Database Link

A complete or partial database link to create a synonym for a schema object on a remote database where the object is located.

#### Object Schema

The schema in which the object resides.


#### Object Type

The object type.

#### Object

The object for which the synonym is created.

## Oracle Triggers

Triggers are similar to procedures. A trigger stored in the database can include SQL and PL/SQL or Java statements to run as a unit and can invoke procedures. Click  -> **Trigger** to open an object list for **Trigger**.

See [Triggers](#) for details.

You can choose the **Trigger Type**: TABLE, VIEW, SCHEMA or DATABASE.

### General Properties for Table Trigger

#### Enable

An enabled trigger runs its trigger action if a triggering statement is issued and the trigger restriction (if any) evaluates to true.

#### Table schema

The table schema of the trigger.

#### Table name

The table you wish to create the trigger.

#### Compound

A compound trigger is a single trigger on a table that allows you to specify actions for each of four timing points:

Timing Point	Section
Before the triggering statement executes	BEFORE STATEMENT
After the triggering statement executes	AFTER STATEMENT
Before each row that the triggering statement affects	BEFORE EACH ROW
After each row that the triggering statement affects	AFTER EACH ROW

**Note:** Support from Oracle 11g or later and you can edit the SQL in Trigger Definition.

### Fire

When defining a trigger, you can specify the trigger timing - whether the trigger action is to be run **BEFORE** or **AFTER** the triggering statement.

### For Each

Oracle Database fires a **ROW** trigger once for each row that is affected by the triggering statement and fires a **STATEMENT** trigger only once when the triggering statement is issued if the optional trigger constraint is met.

### When

To specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger.

### Insert

The trigger is activated whenever adding a row to a table or adds an element to a nested table.

### Delete

The trigger is activated whenever removing a row from the table or removes an element from a nested table.

### Update

The trigger is activated whenever changing a value in one of the fields selected in **Update Of Fields**.

## General Properties for View Trigger

### Enable

An enabled trigger runs its trigger action if a triggering statement is issued and the trigger restriction (if any) evaluates to true.

### Table schema

The view schema of the trigger.

### Table name

The view you wish to create the trigger.

### Nested Table Field

To select the nested table field.

### Compound

To specify the Instead Of Trigger.

**Note:** Support from Oracle 11g or later and you can edit the SQL in Trigger Definition.

### **Insert**

The trigger is activated whenever adding a row to a table or adds an element to a nested table.

### **Delete**

The trigger is activated whenever removing a row from the table or removes an element from a nested table.

### **Update**

The trigger is activated whenever changing a value in a row.

## **General Properties for Schema Trigger**

### **Enable**

An enabled trigger runs its trigger action if a triggering statement is issued and the trigger restriction (if any) evaluates to true.

### **Table schema**

The table schema of the trigger.

### **Fire**

When defining a trigger, you can specify the trigger timing - whether the trigger action is to be run **BEFORE** or **AFTER** the triggering statement.

### **When**

To specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger.

### **Events**

The kind of statement that activates the trigger.

## **General Properties for Database Trigger**

### **Enable**

An enabled trigger runs its trigger action if a triggering statement is issued and the trigger restriction (if any) evaluates to true.

### **Fire**

When defining a trigger, you can specify the trigger timing - whether the trigger action is to be run **BEFORE** or **AFTER** the triggering statement.

### **When**

To specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger.

### **Events**

The kind of statement that activates the trigger.



## Advanced Properties for Table and View Trigger

### Referencing Old

Correlation names of the old nested table.

### Referencing New

Correlation names of the new nested table.

### Referencing Parent

Correlation names of the parent table.

### Follows


To indicate that the trigger should fire after the specified triggers. Use the **Schema** drop-down list to select the schema name and **Trigger** drop-down list to select the trigger.

**Note:** Support from Oracle 11g or later.

### Definition







You can edit valid SQL or procedure statements in the trigger definition inside *BEGIN* and *END*.

## Oracle Types



Type is an user-defined datatype that model the structure and behavior of the data in an application. An object type consists of two parts: a specification and a body. The type body always depends on its type specification. A collection type is a named varying array (varray) or a nested table type. Click  -> **Type** to open an object list for **Type**.

The **Code Outline** window displays information about the object type/object type body including declaration, etc. To show the **Code Outline** window, simply choose **View** -> **Code Outline**.

**Note:** Available only in Full Version.

Button	Description
	Refresh the code outline.
	Show the detail view of the code outline.
	Turn mouse over highlight on or off.
	Expand the selected item.
	Collapse the selected item.
	Toggle sorting by position.

### Object Type's Definition

Enter the object type's definition. After saving the object type, you can edit the Object Type Body. Just click  **New Object Type Body** or  **Design Object Type Body** to open the Type Body Designer.

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

## Object Type Body's Definition

Enter the object type body's definition. To edit the Object Type Specification, click  **Design Object Type Specification** to open the Object Type Designer.

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

## Collection Type

### Nested table

Create a nested table type.

### VArray

Create a varray type.

### Array Size

Determine the array size of the varray type.


### Data Type

Select the Oracle Database built-in datatype or user-defined type of the attribute.

### Data Type Parameter

Determine the corresponding data type parameters.

## Oracle XML Schemas

XML Schema is a schema definition language written in XML. It can be used to describe the structure and various other semantics of conforming instance documents. Click  -> **XML Schema** to open an object list for **XML Schema**.

## Schema Doc

Enter a valid XML schema document under the **Schema Doc** tab.

## Advanced Properties

### Local

Check this to register as local schemas.

### Force on schema registration

Check this to ignore errors generated during schema evolution.

### Object Types

Check this to enable the schema compiler to generate object types.

## Java Beans

Check this to enable the schema compiler to generate Java beans.

## Default Tables

Check this to enable the schema compiler to generate default tables.

## REGISTER\_NODOCID

Check this to prevent the creation of this column if the user wishes to optimize on storage.

## REGISTER\_BINARYXML

Check this to register the schema for Binary XML.

## REGISTER\_NT\_AS\_IOT

Check this to store nested tables created during schema registration as index organized tables.


## REGISTER\_AUTO\_OOL

Check this to automatically move large types out of line.


## Enable Hierarchy

ENABLE_HIERARCHY_NONE	Enable hierarchy will not be called on any tables created while registering that schema.
ENABLE_HIERARCHY_CONTENTS	Enable hierarchy will be called for all tables created during schema registration with hierarchy_type as DBMS_XDBZ.ENABLE_CONTENTS.
ENABLE_HIERARCHY_RESMETADATA	Enable hierarchy will be called on all tables created during schema registration with hierarchy_type as DBMS_XDBZ.ENABLE_RESMETADATA.

## Oracle Recycle Bin


Recycle bin is actually a data dictionary table containing information about dropped objects. Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space. They continue to count against user space quotas, until specifically purged from the recycle bin or the unlikely situation where they must be purged by the database because of tablespace space constraints. Click  -> **Recycle Bin** to open an object list for **Recycle Bin**.

To restore a table

1. Choose a table in recycle bin.
2. Click  **Flashback Table** from the object list toolbar.

To remove an object

1. Select an object for purging in the Object List pane.

2. Click  **Purge Object** from the object list toolbar.
3. Confirm deleting in the dialog window.


To remove all objects

1. Right-click and select the **Purge Recycle Bin** from the pop-up menu.
2. Confirm deleting in the dialog window.

To remove all objects of any users

1. Log in a user has the **SYSDBA** privilege.
2. Right-click and select **Purge DBA Recycle Bin** from the pop-up menu.
3. Confirm deleting in the dialog window.

## Oracle Directories


A directory object specifies an alias for a directory on the server file system where external binary file LOBs (BFILEs) and external table data are located. All directories are created in a single namespace and are not owned by an individual schema. Click  -> **Directory** to open an object list for **Directory**.

### General Properties

#### Directory Path

Specify the full path name of the operating system directory of the server where the files are located. The path name is case sensitive.

## Oracle Tablespaces

Tablespaces are the allocation of space in the database that can contain schema objects. Click  -> **Tablespace** to open an object list for **Tablespace**.

### General Properties

#### Tablespace Type

PERMANENT	A permanent tablespace contains persistent schema objects. Objects in permanent tablespaces are stored in datafiles.
TEMPORARY	A temporary tablespace contains schema objects only for the duration of a session. Objects in temporary tablespaces are stored in tempfiles.
UNDO	An undo tablespace is a type of permanent tablespace used by Oracle Database to manage undo data if you are running your database in automatic undo management mode.

#### Name

Set the name of the datafile/tempfile.

## Size

Set the size of the datafile/tempfile.

## Unit

Define the size unit of the datafile/tempfile. Specify the maximum disk space allowed for automatic extension of the datafile. Use the drop-down list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes.

## Reuse

To allow Oracle to reuse an existing file.

## Path

Specify the path of the datafile/tempfile.

## Auto Extend

To **ON** (enable) or **OFF** (disable) the automatic extension of a new or existing datafile or tempfile.

## Next Size

Specify the size in bytes of the next increment of disk space to be allocated automatically when more extents are required. The default is the size of one data block. Use the drop-down list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes.

## Unlimited Max

Unlimited disk space that Oracle can allocate to the datafile or tempfile.

## Max Size

Specify the maximum disk space allowed for automatic extension of the datafile. Use the drop-down list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes.

## Storage

### File Type

BIGFILE	A bigfile tablespace contains only one datafile or tempfile, which can contain up to approximately 4 billion ( $2^{32}$ ) blocks. The maximum size of the single datafile or tempfile is 128 terabytes (TB) for a tablespace with 32K blocks and 32TB for a tablespace with 8K blocks.
SMALLFILE	A smallfile tablespace is a traditional Oracle tablespace, which can contain 1022 datafiles or tempfiles, each of which can contain up to approximately 4 million ( $2^{22}$ ) blocks.

### Min Extent Size

The minimum size of an extent in the tablespace. Use the drop-down list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes.

### Block Size

The block size for the tablespace.

## Default Storage Options

Set the [default storage characteristics](#) for objects created in the tablespace.

## Table Compression

Use the drop-down list to select the type of compressing data segments to reduce disk use.

## Manual Segment Management

To manage the free space of segments in the tablespace using free lists.

## Extent Management

DICTIONARY	Extent management by the data dictionary.
LOCAL	Extent management by the bitmaps.

## Local Extent

AUTOALLOCATE	The tablespace is system managed.
UNIFORM	The tablespace is managed with uniform extents of size.

## Uniform Size

The size of uniform extent. The default size is 1 megabyte. Use the drop-down list K, M, G, T, P or E to specify the size in kilobytes, megabytes, gigabytes, terabytes, petabytes, or exabytes.

## Advanced Properties

### Logging

LOGGING	Log all objects within the tablespace in the redo log file.
NOLOGGING	No operations are logged.

### Force Logging

Oracle Database will log all changes to all objects in the tablespace except changes to temporary segments, overriding any NOLOGGING setting for individual objects.

### Offline

The tablespace is unavailable immediately (offline) after creation.

### Retention Guarantee

Oracle Database should preserve unexpired undo data in all undo segments of tablespace even if doing so forces the failure of ongoing operations that need undo space in those segments.

### Tablespace Group

To determine whether tablespace is a member of a tablespace group.

### Flashback

ON	Oracle Database will save Flashback log data for this tablespace and the tablespace can participate in a FLASHBACK DATABASE operation.
OFF	Oracle Database will not save any Flashback log data for this tablespace.


### Use Encryption

Enable the encryption properties of the tablespace.

### Algorithm


To select the encryption algorithm.

## Oracle Public Database Links

Public database Link is a database link created by a *DBA* on a local database that is accessible to all users on that database. Click  -> **Public Database Link** to open an object list for **Public Database Link**.

See [Database Link](#) for details.

## Oracle Public Synonyms

Public synonym is a synonym owned by the special user group named *PUBLIC* and every user in a database can access it. Click  -> **Public Synonyms** to open an object list for **Public Synonyms**.

See [Synonyms](#) for details.

## PostgreSQL Objects

To start working with the server objects, you should create and open a connection. If the server is empty, you need to right-click the connection in the Connection pane and choose **New Database** to create a new database.

### General Properties

To create a database, you must have the **Can create database** (usecreatedb) right. Refer to [Role Designer](#) or [User Designer](#) on how to set user properties.

### Database Name

Define the name of the database.

### Encoding

Define the encoding for the database. If omitted, the default is the encoding of the template database.

### Owner

Define the owner for the database. If omitted, defaults to the user executing the command. Only superusers may create database owned by users other than themselves.

## Template

Create the database from a template database.

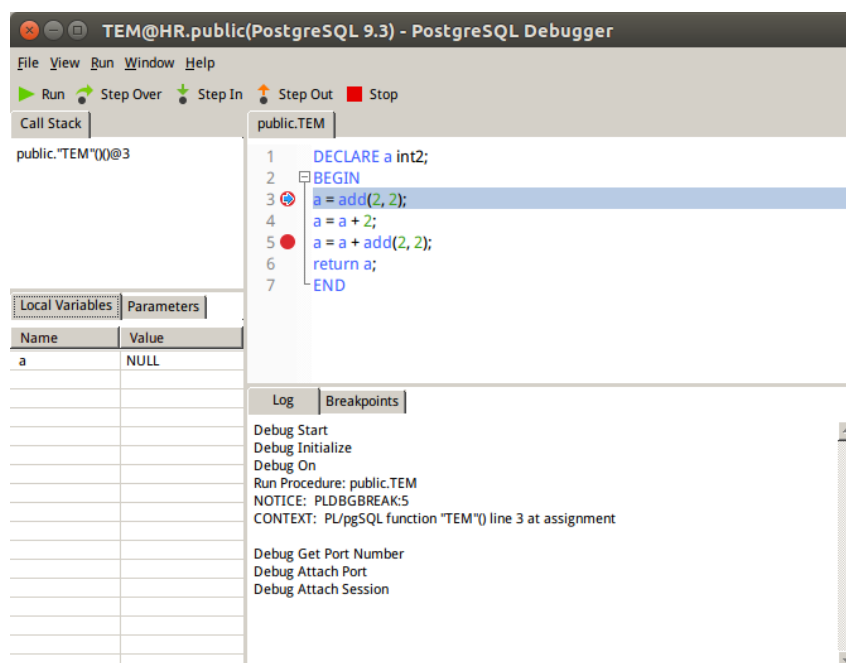
**Note:** It is essential that the source database be idle (no data-altering transactions in progress) for the duration of the copying operation. CREATE DATABASE will check that no session (other than itself) is connected to the source database at the start of the operation, but this does not guarantee that changes cannot be made while the copy proceeds, which would result in an inconsistent copied database. Therefore, it is recommended that databases used as templates be treated as read-only.

## Tablespace




Define the tablespace for the database. If omitted, defaults to pg\_default.

## PostgreSQL Debugger (Available only in Full Version)



Navicat provides PostgreSQL debugger for debugging PostgreSQL PL/pgSQL functions.




You can perform the most commonly used actions for debugging on the toolbar or menu:

Button	Description
 Run	Start running code in debug mode. Enter the Input Parameters if necessary. The debugger executes your code until the end of the code or the next breakpoint is reached. Keyboard shortcut: F9
 Step Over	Resume the execution. The current line will be executed. If the line is a function call, it will bypass the function. The counter will then move to the next line of code. Keyboard shortcut: F8
 Step In	Resume the execution. The current line will be executed. If the line is a function call, the counter goes to the first statement in the function. Otherwise, the counter will move to the next line of code. Keyboard shortcut: F7



 Step Out	Resume the execution. The remaining part of the code within the current function will be executed. Keyboard shortcut: SHIFT+F7
 Stop	Stop stepping the code. The execution will stop and cannot resume it.

The **Code** pane shows the code of the function. You can add/remove breakpoints for debugging by clicking  in the grey area beside each statement.

The **Breakpoints** tab displays all the breakpoints. You can delete a breakpoint or all breakpoints, right-click a breakpoint and choose **Remove Breakpoint** or **Remove All Breakpoints**. To jump to the line of a breakpoint, right-click it and choose **Goto Function**.

The **Call Stack** tab displays the function calls of the current line.

The **Results** tab displays the results after the function has completed the execution.

The **Log** tab shows the message log when debugging the code.

The **Local Variables** tab displays the variables in function.

The **Parameters** tab displays the inputted parameters.

## PostgreSQL Schemas

A schema is essentially a namespace: it contains named objects (tables, data types, functions, and operators) whose names may duplicate those of other objects existing in other schemas. The schema name must be distinct from any existing schema name in the current database.

### General Properties


#### Schema Name



The name of a schema to be created. The name cannot begin with pg\_, as such names are reserved for system schemas.



#### Owner

The name of the user who will own the schema. If omitted, defaults to the user executing the command.

## PostgreSQL Tables

Relational databases use tables to store data. All operations on data are done on the tables themselves or produce another table as the result. A table is a set of rows and columns, and their intersections are fields. From a general perspective, columns within a table describe the name and type of data that will be found by row for that column's fields. Rows within a table represent records composed of fields that are described from left to right by their corresponding column's name and type. Each field in a row is implicitly correlated with each other field in that row. Click  to open an object list for **Table**.

When open a table with graphical fields, right-click a table and select the  **Open Table (Quick)** from the pop-up menu. Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell. If you do wish Navicat loads all your images while opening the table, use  **Open Table**.

To create a new table, click the  **New Table** from the object list toolbar together with the  down arrow to choose the type **Normal / Foreign**.







To empty a table, right-click the selected table and choose **Empty Table** from the pop-up menu. This option is only applied when you wish to clear all the existing records without resetting the auto-increment value. To reset the auto-increment value while emptying your table, use **Truncate Table**.

## PostgreSQL Normal Tables

Tables are the basic unit of data storage in a PostgreSQL database. Data is stored in rows and columns. You define a table with a table name and set of columns.

## PostgreSQL Table Fields

In the **Fields** tab, just simply click a field for editing. A right-click displays the pop-up menu or using the field toolbar, you can create new and drop the selected field. To search a field name, choose **Edit -> Find** or press CTRL+F.

Button	Description
 Add Field	Add a field to the table.
 Insert Field	Insert a field above an existing field. <b>Note:</b> Only available when creating a new table.
 Delete Field	Delete the selected field.
 Primary Key	Set the field as a Primary Key and set the <b>Primary Key Properties</b> .
 Move Up/ Down  Move Down	Move the selected field up/down. <b>Note:</b> Only available when creating a new table.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** drop-down list defines the type of the field data. See [PostgreSQL Data Types](#) for details.

Use the **Length** edit box to define the length of the field and use **Decimals** edit box to define the number of digits after the decimal point (the scale) for Floating Point data type.

**Note:** Be careful when shortening the field length as losing data might be caused.

### Not null

Check this box to not allow the NULL values for the field.

## Primary Key

A Primary Key is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.

### Field's Pop-up Options

Duplicate Field	Add a new field with modification as one of the existing fields.
Primary Key Properties	<p><b>Name</b> The primary key constraint name.</p> <p><b>Tablespace</b> The tablespace of the primary key constraint's index.</p> <p><b>Fill Factor</b> The fillfactor storage parameter of the primary key constraint's index.</p> <p><b>Deferrable</b> The primary key constraint can be deferred. <b>Note:</b> Support from PostgreSQL 9.0 or later.</p> <p><b>Deferred</b> The primary key constraint is checked only at the end of the transaction. <b>Note:</b> Support from PostgreSQL 9.0 or later.</p>

### Field's Properties

**Note:** The following options depend on the field type you are chosen.

#### Default

Set the default value for the field.

#### Comment

Set any optional text describing the current field.

#### Collation

Set the collation of the column (which must be of a collatable data type). If not specified, the column data type's default collation is used.

**Note:** Support from PostgreSQL 9.1 or later.

#### Dimensions

Set the dimensions of array specifiers.

#### Object Schema

Set the object schema for the field.



## Object Type

Set the object type for the field.


## PostgreSQL Table Indexes

Indexes are primarily used to enhance database performance (though inappropriate use can result in slower performance). An index field can be an expression computed from the values of one or more columns of the table row. This feature can be used to obtain fast access to data based on some transformation of the basic data.

In the **Indexes** tab, just simply click an index field for editing. By using the index toolbar, you can create new, edit and delete the selected index field.

Button	Description
 Add Index	Add an index to the table.
 Delete Index	Delete the selected index.

Use the **Name** edit box to set the index name. No schema name can be included here; the index is always created in the same schema as its parent table.

To include field(s) in the index, just simply double-click the **Fields** field or click  to open the editor for editing.

**Note:** Some of field types do not allow indexing by several fields.

The **Index method** drop-down list defines the type of the table index.

### Unique

Makes index unique, causes the system to check for duplicate values in the table when the index is created (if data already exist) and each time data is added.

### Clustered

*CLUSTER* instructs PostgreSQL to cluster the table specified by tablename based on the index specified by indexname. The index must already have been defined on tablename.

When a table is clustered, PostgreSQL remembers on which index it was clustered. The form *CLUSTER tablename* reclusters the table on the same index that it was clustered before.

### Concurrently

When this option is used, PostgreSQL will build the index without taking any locks that prevent concurrent inserts, updates, or deletes on the table; whereas a standard index build locks out writes (but not reads) on the table until it's done.

### Tablespace

The tablespace in which to create the index.

### Fill Factor

The fillfactor for an index.

### Buffering

Use the buffering build technique to build the index.

**Note:** Support from PostgreSQL 9.2 or later.

### Fast Update

The usage of the fast update technique.

**Note:** Support from PostgreSQL 8.4 or later.

### Constraints

If you wish to create partial index, enter constraint condition in this edit box. A partial index is an index that contains entries for only a portion of a table, usually a portion that is more useful for indexing than the rest of the table.

### Comment

Define the comment for the index.

### Fields Editor

Select the field(s) from the **Name** list. You can also use the arrow buttons to change the index field(s) order.

### Collation

Choose the collation for the index.

**Note:** Support from PostgreSQL 9.1 or later.

### Sort Order

Specify the sort order: ASC or DESC.


### Nulls Order


Specify that nulls sort before(NULLS FIRST)/after(NULLS LAST) non-nulls.

## PostgreSQL Table Foreign Keys

A foreign key specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table. We say this maintains the referential integrity between two related tables.


In the **Foreign Keys** tab, just simply click a foreign key field for editing. By using the foreign key toolbar, you can create new, edit and delete the selected foreign key field.

Button	Description
 Add Foreign Key	Add a foreign key to the table.

 Delete Foreign Key	Delete the selected foreign key.
------------------------------------------------------------------------------------------------------	----------------------------------

Use the **Name** edit box to enter a name for the new key.

Use the **Referenced Schema** and **Referenced Table** drop-down lists to select a foreign schema and table respectively.

To include field(s)/referenced field(s) to the key, just simply double-click the **Fields/Referenced Fields** field or click  to open the editor(s) for editing.

The **On Delete** and **On Update** drop-down list define the type of the actions to be taken.

RESTRICT	Produce an error indicating that the deletion or update would create a foreign key constraint violation. This is the same as NO ACTION except that the check is not deferrable.
NO ACTION	Produce an error indicating that the deletion or update would create a foreign key constraint violation. If the constraint is deferred, this error will be produced at constraint check time if there still exist any referencing rows. This is the default action.
CASCADE	Delete any rows referencing the deleted row, or update the value of the referencing column to the new value of the referenced column, respectively.
SET NULL	Set the referencing column(s) to null.
SET DEFAULT	Set the referencing column(s) to their default values.

Related topic:

[Foreign Keys Data Selection](#)

### Match Full

Check this option to not allow one column of a multicolumn foreign key to be null unless all foreign key columns are null.

### Deferrable

The foreign key constraint can be deferred.

### Deferred

The foreign key constraint is checked only at the end of the transaction.



### Comment

Define the comment for the foreign key.

## PostgreSQL Table Uniques

Unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

In the **Uniques** tab, just simply click an unique field for editing. By using the unique toolbar, you can create new, edit and delete the selected unique field.

Button	Description
 Add Unique	Add an unique to the table.
 Delete Unique	Delete the selected unique.

Use the **Name** edit box to set the unique name.

To set field(s) as unique, just simply double-click the **Fields** field or click  to open the editor(s) for editing.

### Tablespace

The tablespace of the unique constraint's index.

### Fill Factor

The fillfactor storage parameter of the unique constraint's index.

**Note:** Support from PostgreSQL 8.2 or later.

### Deferrable

The unique constraint can be deferred.

### Deferred

The unique constraint is checked only at the end of the transaction.



### Comment

Define the comment for the unique.

## PostgreSQL Table Checks

A check constraint is the most generic constraint type. It allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

In the **Checks** tab, just simply click a check field for editing. By using the check toolbar, you can create new, edit and delete the selected check field.

Button	Description
 Add Check	Add a check to the table.
 Delete Check	Delete the selected check.

Use the **Name** edit box to set the check name.

### Check

Set the condition for checking, e.g. "field\_name1 > 0 AND field\_name2 > field\_name1" in the **Check** edit box. A check constraint specified as a column constraint should reference that column's value only, while an expression appearing in a table constraint may reference multiple columns.

### Definition

Enter the definition for the check.

### No Inherit

The check constraint will not propagate to child tables.

**Note:** Support from PostgreSQL 9.2 or later.

### Comment



Enter the comment for the check.

## PostgreSQL Table Excludes

A exclude constraint guarantees that if any two rows are compared on the specified column(s) or expression(s) using the specified operator(s), not all of these comparisons will return TRUE.

In the **Excludes** tab, just simply click an exclude field for editing. By using the exclude toolbar, you can create new, edit and delete the selected exclude field.

**Note:** Exclude is supported from PostgreSQL 9.0 or later.

Button	Description
 Add Exclude	Add an exclude to the table.
 Delete Exclude	Delete the selected exclude.

Use the **Name** edit box to set the exclude name.

### Index method

The name of the index access method to be used.

### Element

Choose the element(s) to be excluded and specify the operator(s).

### Tablespace

The tablespace of the exclude constraint's index.

### Fill Factor

The fillfactor storage parameter of the exclude constraint's index.

### Predicate

Specify an exclusion constraint on a subset of the table.

### Buffering

Use the buffering build technique to build the exclude constraint's index.



**Note:** Support from PostgreSQL 9.2 or later.

### **Predicate**

Specify an exclusion constraint on a subset of the table.

### **Deferrable**

The exclude constraint can be deferred.

### **Deferred**

The exclude constraint is checked only at the end of the transaction.

### **Comment**



Define the comment for the exclude.

## PostgreSQL Table Rules

The PostgreSQL rule system allows one to define an alternate action to be performed on insertions, updates, or deletions in database tables. Roughly speaking, a rule causes additional commands to be executed when a given command on a given table is executed.

**Note:** You must be the owner of a table to create or change rules for it.

In the **Rules** tab, just simply click a rule field for editing. By using the rule toolbar, you can create new, edit and delete the selected rule field.

Button	Description
 Add Rule	Add a rule to the table.
 Delete Rule	Delete the selected rule.

Use the **Name** edit box to set the rule name. This must be distinct from the name of any other rule for the same table. Multiple rules on the same table and same event type are applied in alphabetical name order.

### **Event**

The event is one of *SELECT*, *INSERT*, *UPDATE*, or *DELETE*.

### **Do instead**

This indicates that the commands should be executed instead of the original command. Otherwise, the commands should be executed in addition to the original command.

### **Condition**

Any SQL conditional expression (returning boolean). The condition expression may not refer to any tables except NEW and OLD, and may not contain aggregate functions.

## Definition

The command or commands that make up the rule action. Valid commands are *SELECT*, *INSERT*, *UPDATE*, *DELETE*, or *NOTIFY*.

Within condition and command, the special table names NEW and OLD may be used to refer to values in the referenced table. NEW is valid in ON INSERT and ON UPDATE rules to refer to the new row being inserted or updated. OLD is valid in ON UPDATE and ON DELETE rules to refer to the existing row being updated or deleted.

## Comment





Define the comment for the rule.

## PostgreSQL Table Triggers

A trigger is a specification that the database should automatically execute a particular function whenever a certain type of operation is performed. Triggers can be defined to execute either before or after any INSERT, UPDATE, or DELETE operation, either once per modified row, or once per SQL statement.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

**Note:** To create a trigger on a table, the user must have the [TRIGGER](#) privilege on the table.

Button	Description
  Add Trigger	Add a trigger to the table.
  Delete Trigger	Delete the selected trigger.

Use the **Name** edit box to set the trigger name. This must be distinct from the name of any other trigger for the same table.

## Row trigger

This specifies whether the trigger procedure should be fired once for every row affected by the trigger event, or just once per SQL statement. If unchecked, *FOR EACH STATEMENT* is the default.

## Fires

Define the trigger action time. It can be **Before** or **After** to indicate that the trigger activates before or after the statement that activated it.

## Insert

The trigger is activated whenever a new row is inserted into the table.

## Update

The trigger is activated whenever a row is modified.

## Delete

The trigger is activated whenever a row is deleted from the table.

### **Update Of Fields**

Specify a list of columns. The trigger will only fire if at least one of the listed columns is mentioned as a target of the UPDATE command.

**Note:** Support from PostgreSQL 9.1 or later.

### **When Clause**

Specify a Boolean WHEN condition, which will be tested to see whether the trigger should be fired.

**Note:** Support from PostgreSQL 9.0 or later.

### **Trigger Function Schema and Trigger Function**

A user-supplied function that is declared as taking no arguments and returning type trigger, which is executed when the trigger fires.

### **Arguments**

An optional comma-separated list of arguments to be provided to the function when the trigger is executed. The arguments are literal string constants. Simple names and numeric constants may be written here, too, but they will all be converted to strings. Please check the description of the implementation language of the trigger function about how the trigger arguments are accessible within the function; it may be different from normal function arguments.

### **Comment**

Define the comment for the trigger.

### **Constraint**

Create a constraint trigger.

### **Deferrable**

The trigger constraint can be deferred.

### **Deferred**

The trigger constraint is checked only at the end of the transaction.

### **Referenced Table Schema and Referenced Table Name**

The schema and the name of another table referenced by the constraint.

## **PostgreSQL Table Options**

### **Unlogged**

The table is created as an unlogged table. Data written to unlogged tables is not written to the write-ahead log, which makes them considerably faster than ordinary tables.

**Note:** Support from PostgreSQL 9.1 or later.

### Owner

Define the user to own this table.


### Tablespace

Define a tablespace different from the default tablespace to create a table.

**Note:** Support from PostgreSQL 8.0 or later.

### Inherits from

This option specifies a list of tables from which the new table automatically inherits all columns. Use of inheritance creates a persistent relationship between the new child table and its parent table(s). Schema modifications to the parent(s) normally propagate to children as well, and by default the data of the child table is included in scans of the parent(s).

To set the new table to be inherited from one or several existing tables, just simply click  to open the editor(s) for editing.

### Has Oids

Check this option if you want to specify whether rows of the new table should have OIDs (object identifiers) assigned to them.

### Fill Factor

The fillfactor for a table is a percentage between 10 and 100. 100 (complete packing) is the default. When a smaller fillfactor is specified, INSERT operations pack table pages only to the indicated percentage; the remaining space on each page is reserved for updating rows on that page. This gives UPDATE a chance to place the updated copy of a row on the same page as the original, which is more efficient than placing it on a different page. For a table whose entries are never updated, complete packing is the best choice, but in heavily updated tables smaller fillfactors are appropriate.

**Note:** Support from PostgreSQL 8.2 or later.

## PostgreSQL Foreign Tables






Foreign tables define the structure of the remote data. A foreign table can be used in queries just like a normal table, but a foreign table has no storage in the PostgreSQL server. Whenever it is used, PostgreSQL asks the foreign data wrapper to fetch data from the external source, or transmit data to the external source in the case of update commands.

**Note:** Support from PostgreSQL 9.1 or later.

## Fields for PostgreSQL Foreign Tables

In the **Fields** tab, just simply click a field for editing. A right-click displays the pop-up menu or using the field toolbar, you can create new and drop the selected field. To search a field name, choose **Edit -> Find** or press CTRL+F.

Button	Description
--------	-------------

 Add Field	Add a field to the table.
 Insert Field	Insert a field above an existing field. <b>Note:</b> Only available when creating new table.
 Delete Field	Delete the selected field.
 Move Up/ Down  Move Down	Move the selected field up/down. <b>Note:</b> Only available when creating new table.

To add a new field with modification as one of the existing fields, you can right-click and select the **Duplicate Field** from the pop-up menu.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** drop-down list defines the type of the field data. See [PostgreSQL Data Types](#) for details.

Use the **Length** edit box to define the length of the field and use **Decimals** edit box to define the number of digits after the decimal point (the scale) for Floating Point data type.

**Note:** Be careful when shortening the field length as losing data might be caused.

### Not null

Check this box to not allow the NULL values for the field.

### Field's Properties

**Note:** The following options depend on the field type you are chosen.

### Options

Options to be associated with the foreign table column. The allowed option **Name** and **Value** are specific to each foreign data wrapper and are validated using the foreign-data wrapper's validator function.

### Default

Set the default value for the field.

### Comment

Set any optional text describing the current field.

### Collation

Set the collation of the column (which must be of a collatable data type). If not specified, the column data type's default collation is used.

**Note:** Support from PostgreSQL 9.1 or later.

### Dimensions

Set the dimensions of array specifiers.

## Object Schema

Set the object schema for the field.

## Object Type



Set the object type for the field.

## Checks for PostgreSQL Foreign Tables

A check constraint is the most generic constraint type. It allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

In the **Checks** tab, just simply click a check field for editing. By using the check toolbar, you can create new, edit and delete the selected check field.

**Note:** Support from PostgreSQL 9.5 or later.

Button	Description
 Add Check	Add a check to the table.
 Delete Check	Delete the selected check.

Use the **Name** edit box to set the check name.

## Check

Set the condition for checking, e.g. "field\_name1 > 0 AND field\_name2 > field\_name1" in the **Check** edit box. A check constraint specified as a column constraint should reference that column's value only, while an expression appearing in a table constraint may reference multiple columns.

## Definition

Enter the definition for the check.

## No Inherit

The check constraint will not propagate to child tables.

## Comment

Enter the comment for the check.



## Triggers for PostgreSQL Foreign Tables

A trigger is a specification that the database should automatically execute a particular function whenever a certain type of operation is performed. Triggers can be defined to execute either before or after any INSERT, UPDATE, or DELETE operation, either once per modified row, or once per SQL statement.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

**Note:** Support from PostgreSQL 9.4 or later.

**Note:** To create a trigger on a table, the user must have the [TRIGGER](#) privilege on the table.

Button	Description
 Add Trigger	Add a trigger to the table.
 Delete Trigger	Delete the selected trigger.

Use the **Name** edit box to set the trigger name. This must be distinct from the name of any other trigger for the same table.

### Row trigger

This specifies whether the trigger procedure should be fired once for every row affected by the trigger event, or just once per SQL statement. If unchecked, *FOR EACH STATEMENT* is the default.

### Fires

Define the trigger action time. It can be **Before** or **After** to indicate that the trigger activates before or after the statement that activated it.

### Insert

The trigger is activated whenever a new row is inserted into the table.

### Update

The trigger is activated whenever a row is modified.

### Delete

The trigger is activated whenever a row is deleted from the table.

### Update Of Fields

Specify a list of columns. The trigger will only fire if at least one of the listed columns is mentioned as a target of the UPDATE command.

### When Clause

Specify a Boolean WHEN condition, which will be tested to see whether the trigger should be fired.

### Trigger Function Schema and Trigger Function

A user-supplied function that is declared as taking no arguments and returning type trigger, which is executed when the trigger fires.

### Arguments

An optional comma-separated list of arguments to be provided to the function when the trigger is executed. The arguments are literal string constants. Simple names and numeric constants may be written here, too, but they will all be converted to strings. Please check the description of the implementation language of the trigger function about how the trigger arguments are accessible within the function; it may be different from normal function arguments.

## Comment

Define the comment for the trigger.

## Table Options for PostgreSQL Foreign Tables

### Foreign Server

The name of an existing server for the foreign table.

**Note:** Support from PostgreSQL 9.1 or later.


### Options

Options to be associated with the foreign table. The allowed option **Name** and **Value** are specific to each foreign data wrapper and are validated using the foreign-data wrapper's validator function.

### Owner

Define the user to own this table.

### Inherits from

This option specifies a list of tables from which the new table automatically inherits all columns. To set the new table to be inherited from one or several existing tables, just simply click  to open the editor for editing.


**Note:** Support from PostgreSQL 9.5 or later.




### Has Oids

Check this option if you want to specify whether rows of the new table should have OIDs (object identifiers) assigned to them.

**Note:** Support from PostgreSQL 9.5 or later.

## PostgreSQL Views

Views are useful for allowing users to access a set of relations (tables) as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table). Click  to open an object list for **View**.

Button	Description
 Preview	Preview the result of the view.
 Explain	Show the Query Plan of the view.
 Beautify SQL	Format the codes with the Beautify SQL settings in Editor.

**Note:** You can choose to show the Result tab below the editor or in a new tab by selecting **View -> Show Result -> Below Query Editor** or **In a New Tab**.

### View Builder (Available only in Full Version)



**View Builder** allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See [Query Builder](#) for details.

### Definition

You can edit the view definition as SQL statement (SELECT statement it implements).

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

### Rules

Use the **Name** edit box to set the rule name.

#### Event

The event is one of *SELECT*, *INSERT*, *UPDATE*, or *DELETE*.

#### Do instead

This indicates that the commands should be executed instead of the original command. Otherwise, the commands should be executed in addition to the original command.

#### Condition

Any SQL conditional expression (returning boolean). The condition expression may not refer to any tables except NEW and OLD, and may not contain aggregate functions.

#### Definition

The command or commands that make up the rule action. Valid commands are *SELECT*, *INSERT*, *UPDATE*, *DELETE*, or *NOTIFY*.

Within condition and command, the special table names NEW and OLD may be used to refer to values in the referenced table. NEW is valid in ON INSERT and ON UPDATE rules to refer to the new row being inserted or updated. OLD is valid in ON UPDATE and ON DELETE rules to refer to the existing row being updated or deleted

#### Comment

Define the comment for the rule.

### Advanced Properties

#### Owner

The owner of the view.


### View Viewer

**View Viewer** displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See [Table Viewer](#) for details.

## PostgreSQL Functions

PostgreSQL provides four kinds of functions:

- query language functions (functions written in SQL)
- procedural language functions (functions written in, for example, PL/Tcl or PL/pgSQL)
- internal functions
- C-language functions

Every kind of function can take base types, composite types, or combinations of these as arguments (parameters). In addition, every kind of function can return a base type or a composite type. Many kinds of functions can take or return certain pseudo-types (such as polymorphic types), but the available facilities vary. Click  to open an object list for **Function**.

To install the pldbgapi extension for debugging PL/pgSQL functions, you can right-click anywhere in the Object List pane and choose **Install pldbgapi Extension**.

### Function Wizard

Click the  **New Function** from the object list toolbar. The **Function Wizard** will pop up and it allows you to create a function easily. You are allowed not to show the **Function Wizard** when create new function.

1. Define the parameter(s) of the function. Set the parameter **Mode**, **Type Schema**, **Type**, **Name** and **Default Value** under corresponding columns.
2. Select the **Schema** and **Return Type** from the list.

**Hint:** Once uncheck the **Show wizard next time**, you can go to [Options](#) to enable it.

### Definition

Definition consists of a valid SQL procedure statement. This can be a simple statement such as *SELECT* or *INSERT*, or it can be a compound statement written using *BEGIN* and *END*. Compound statements can contain declarations, loops, and other control structure statements.

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

### Parameter

Define function parameter.

### Return type schema and Return Type

The return type of the function.

### Table Columns

Enter the table columns if the return type is set to TABLE.

## Advanced Properties

### Owner

The owner of the function.

**Note:** Support from PostgreSQL 8.0 or later.

### Language

The name of the language that the function is implemented in. May be SQL, C, internal, or the name of a user-defined procedural language. For backward compatibility, the name may be enclosed by single quotes.

### Volatility

These attributes inform the query optimizer about the behavior of the function. At most one choice may be specified. If none of these appear, VOLATILE is the default assumption.

IMMUTABLE	The function cannot modify the database and always returns the same result when given the same argument values; that is, it does not do database lookups or otherwise use information not directly present in its argument list. If this option is given, any call of the function with all-constant arguments can be immediately replaced with the function value.
STABLE	The function cannot modify the database, and that within a single table scan it will consistently return the same result for the same argument values, but that its result could change across SQL statements. This is the appropriate selection for functions whose results depend on database lookups, parameter variables (such as the current time zone), etc. Also note that the current_timestamp family of functions qualify as stable, since their values do not change within a transaction.
VOLATILE	The function value can change even within a single table scan, so no optimizations can be made. Relatively few database functions are volatile in this sense; some examples are random(), curval(), timeofday(). But note that any function that has side-effects must be classified volatile, even if its result is quite predictable, to prevent calls from being optimized away; an example is setval().

### Security of definer

Specify that the function is to be executed with the privileges of the user that created it.

### Returns Set

Indicate that the function will return a set of items, rather than a single item.

### Strict

Indicate that the function always returns null whenever any of its arguments are null. If this parameter is specified, the function is not executed when there are null arguments; instead a null result is assumed automatically.

### Estimated cost

A positive number giving the estimated execution cost for the function, in units of cpu\_operator\_cost. If the function returns a set, this is the cost per returned row. If the cost is not specified, 1 unit is assumed for C-language and internal

functions, and 100 units for functions in all other languages. Larger values cause the planner to try to avoid evaluating the function more often than necessary.

**Note:** Support from PostgreSQL 8.3 or later.

### Estimated rows

A positive number giving the estimated number of rows that the planner should expect the function to return. This is only allowed when the function is declared to return a set.


**Note:** Support from PostgreSQL 8.3 or later.

### Configuration parameter

The specified configuration parameter to be set to the specified value when the function is entered, and then restored to its prior value when the function exits.

**Note:** Support from PostgreSQL 8.3 or later.

### Result


To run the function, click  **Run** on the toolbar. If the SQL statement is correct, the statement will be executed and, if the statement is supposed to return data, the **Result** tab opens with the data returned by the function. If an error occurs while executing the function, execution stops, the appropriate error message is displayed. If the function requires input parameter, the **Input Parameters** box will pop up. Use ',' to separate the parameters.

**Hint:** Navicat supports to return 10 resultsets.

### Debug (Available only in Full Version)

To debug the PL/pgSQL function, click  **Debug Function** on the toolbar to launch the [PostgreSQL Debugger](#).

## PostgreSQL Aggregates

Aggregate functions in PostgreSQL are expressed as state values and state transition functions. That is, an aggregate can be defined in terms of state that is modified whenever an input item is processed. To define a new aggregate function, one selects a data type for the state value, an initial value for the state, and a state transition function. The state transition function is just an ordinary function that could also be used outside the context of the aggregate. A final function can also be specified, in case the desired result of the aggregate is different from the data that needs to be kept in the running state value. Click  -> **Aggregate** to open an object list for **Aggregate**.

### Properties

#### Owner

The owner of the aggregate function.

**Note:** Support from PostgreSQL 8.0 or later.

### Input type

An input data type on which this aggregate function operates.

**Note:** Support from PostgreSQL 8.2 or later. For versions below 8.2, just select the **Input type schema** and **Input type** from the drop-down lists.

### State Type Schema and State type

The data type for the aggregate's state value.

### State Function Schema and State function

The state transition function to be called for each input row. For an N-argument aggregate function, the state function must take N+1 arguments, the first being of type *state\_data\_type* and the rest matching the declared input data type(s) of the aggregate. The function must return a value of type *state\_data\_type*. This function takes the current state value and the current input data value(s), and returns the next state value.

### Final Function Schema and Final function

The final function called to compute the aggregate's result after all input rows have been traversed. The function must take a single argument of type *state\_data\_type*. The return data type of the aggregate is defined as the return type of this function. If final function is not specified, then the ending state value is used as the aggregate's result, and the return type is *state\_data\_type*.

### Initial condition


The initial setting for the state value. This must be a string constant in the form accepted for the data type *state\_data\_type*. If not specified, the state value starts out null.

### Sort operator schema and Sort operator

The associated sort operator for a MIN- or MAX-like aggregate. The operator is assumed to have the same input data types as the aggregate (which must be a single-argument aggregate).

**Note:** Support from PostgreSQL 8.1 or later.

## PostgreSQL Conversions

Conversion defines a new conversion between character set encodings. Conversion names may be used in the convert function to specify a particular encoding conversion. Also, conversions that are marked DEFAULT can be used for automatic encoding conversion between client and server. For this purpose, two conversions, from encoding A to B and from encoding B to A, must be defined. Click  -> **Conversion** to open an object list for **Conversion**.

### Properties

#### Owner

The owner of the conversion function.

**Note:** Support from PostgreSQL 8.0 or later.

### Source encoding

The source encoding name.

### Target encoding

The destination encoding name.

### Schema of function and Function

The function used to perform the conversion. The function name may be schema-qualified. If it is not, the function will be looked up in the path.


The function must have the following signature:

```
conv_proc(  
integer, -- source encoding ID  
integer, -- destination encoding ID  
cstring, -- source string (null terminated C string)  
internal, -- destination (fill with a null terminated C string)  
integer -- source string length  
) RETURNS void;
```

### Default

Check this box to indicate that this conversion is the default for this particular source to destination encoding. There should be only one default encoding in a schema for the encoding pair.

## PostgreSQL Domains

A domain is essentially a data type with optional constraints (restrictions on the allowed set of values). The user who defines a domain becomes its owner. Domains are useful for abstracting common constraints on fields into a single location for maintenance. For example, several tables might contain email address columns, all requiring the same *CHECK* constraint to verify the address syntax. Define a domain rather than setting up each table's constraint individually. Click  -> **Domain** to open an object list for **Domain**.

### General Properties

#### Underlying Type Category

Choose the underlying data type category: **Base Type**, **Composite Type**, **Enum Type** and **Domain**.

**Note:** Support from PostgreSQL 8.2 or later.

#### Underlying Type Schema

Select schema of the underlying data type.

#### Underlying Type

Select the underlying data type of the domain from the drop-down list.

## Dimensions

The dimensions of array specifiers.

## Length and Scale

Use the **Length** edit box to define the length of the field and use **Scale** edit box to define the number of digits after the decimal point. (if required for the selected data type)

## Collate

The collation of the domain.

**Note:** Support from PostgreSQL 9.1 or later.

## Default

The *DEFAULT* clause specifies a default value for columns of the domain data type. The value is any variable-free expression (but subqueries are not allowed). The data type of the default expression must match the data type of the domain. If no default value is specified, then the default value is the null value.

The default expression will be used in any insert operation that does not specify a value for the column. If a default value is defined for a particular column, it overrides any default associated with the domain. In turn, the domain default overrides any default value associated with the underlying data type.

## Not null

Values of this domain are not allowed to be null.

## Owner

The owner of the domain function. The user who defines a domain becomes its owner.


**Note:** Support from PostgreSQL 7.4 or later.

## Checks

The **Checks** tab is provided for managing domain checks. It allows you to create new, edit, or delete the selected check.

*CHECK* clauses specify integrity constraints or tests which values of the domain must satisfy. Each constraint must be an expression producing a Boolean result. It should use the key word *VALUE* to refer to the value being tested.

## PostgreSQL Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table. Click  -> **Index** to open an object list for **Index**.

## General Properties

**Unique**

Makes index unique, causes the system to check for duplicate values in the table when the index is created (if data already exist) and each time data is added.

**Table name**

The name (possibly schema-qualified) of the table to be indexed.

**Method**

Define the type of the index.

**Name/Expression**

The name of a column of the table. Or, an expression based on one or more columns of the table.

**Collate**

Choose the collation for the index.

**Note:** Support from PostgreSQL 9.1 or later.

**Operator Class Schema and Operator Class**

The schema and name of an operator class.

**Sort Order** (*only for B-Tree index*)

Specify the sort order: ASC or DESC.

**Nulls Order** (*only for B-Tree index*)

Specify that nulls sort before(NULLS FIRST)/after(NULLS LAST) non-nulls.

**Advanced Properties****Clustered**

*CLUSTER* instructs PostgreSQL to cluster the table specified by tablename based on the index specified by indexname. The index must already have been defined on tablename.

When a table is clustered, PostgreSQL remembers on which index it was clustered. The form *CLUSTER tablename* reclusters the table on the same index that it was clustered before.

**Concurrently**

When this option is used, PostgreSQL will build the index without taking any locks that prevent concurrent inserts, updates, or deletes on the table; whereas a standard index build locks out writes (but not reads) on the table until it's done.

**Tablespace**

The tablespace in which to create the index.

**Fill Factor (%)**

The fillfactor for an index is a percentage that determines how full the index method will try to pack index pages.



## Buffering

Use the buffering build technique to build the index.

**Note:** Support from PostgreSQL 9.2 or later.

## Fast Update


This setting controls usage of the fast update technique.

**Note:** Support from PostgreSQL 8.4 or later.

## Constraint




If you wish to create partial index, enter constraint condition in this edit box. A partial index is an index that contains entries for only a portion of a table, usually a portion that is more useful for indexing than the rest of the table.

## PostgreSQL Materialized Views

Materialized view is a view of a query that is physically materialized. The query is executed and used to populate the view at the time the command is issued and may be refreshed later. Click  -> **Materialized View** to open an object list for **Materialized View**.

**Note:** Support from PostgreSQL 9.3 or later.

To refresh and completely replace the contents of a materialized view, right-click it in the Object List pane and select **Refresh Materialized View With -> Data** or **No Data** from the pop-up menu.

Button	Description
 Preview	Preview the result of the materialized view.
 Explain	Show the Query Plan of the materialized view.
 Beautify SQL	Format the codes with the Beautify SQL settings in Editor.

**Note:** You can choose to show the Result tab below the editor or in a new tab by selecting **View -> Show Result -> Below Query Editor** or **In a New Tab**.

## View Builder (Available only in Full Version)

**View Builder** allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See [Query Builder](#) for details.

## Definition

You can edit the view definition as SQL statement (SELECT statement it implements).

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

## Advanced Properties

### Owner

The owner of the materialized view.

### Tablespace

The name of the tablespace in which the new materialized view is to be created.

### Fill Factor

The fillfactor for a view is a percentage between 10 and 100. 100 (complete packing) is the default.


### With Data

The materialized view should be populated at creation time.

### View Viewer

**View Viewer** displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See [Table Viewer](#) for details.

## PostgreSQL Operators

PostgreSQL supports left unary, right unary, and binary operators. Operators can be overloaded. At least one of *LEFTARG* and *RIGHTARG* must be defined. For binary operators, both must be defined. For right unary operators, only *LEFTARG* should be defined, while for left unary operators only *RIGHTARG* should be defined. Click  -> **Operator** to open an object list for **Operator**.

**Note:** *LEFTARG* = Left type; *RIGHTARG* = Right type.

### General Properties

#### Owner

The owner of the operator function.

**Note:** Support from PostgreSQL 8.0 or later.

#### Schema of left type and Left type

The data type of the operator's left operand, if any. This option would be omitted for a left-unary operator.

#### Schema of right type and Right type

The data type of the operator's right operand, if any. This option would be omitted for a right-unary operator.

#### Schema of operator function and Operator function

The function used to implement this operator.

### Advanced Properties

#### Schema of restrict function and Restrict function

The restriction selectivity estimator function for this operator.

### Schema of join function and Join function

The join selectivity estimator function for this operator.

### Schema of commutator and Commutator

The commutator of this operator.

### Schema of negator and Negator

The negator of this operator.

### Hash

The operator can support a hash join if this option on.

### Merge

The operator can support a merge join if this option on.

## Additional Advanced Properties for PostgreSQL Version below 8.3

### Schema of left sort operator and Left sort operator

If this operator can support a merge join, the left sort operator that sorts the left-hand data type of this operator.

### Schema of right sort operator and Right sort operator

If this operator can support a merge join, the right sort operator that sorts the right-hand data type of this operator.

### Schema of less than operator and Less than operator

If this operator can support a merge join, the less-than operator that compares the input data types of this operator.

### Schema of greater than operator and Greater than operator

If this operator can support a merge join, the greater-than operator that compares the input data types of this operator.

## PostgreSQL Operator Classes

An operator class defines how a particular data type can be used with an index. The operator class specifies that certain operators will fill particular roles or "strategies" for this data type and this index method. The operator class also specifies the support procedures to be used by the index method when the operator class is selected for an index column. All the operators and functions used by an operator class must be defined before the operator class is created.

Click  -> **Operator Class** to open an object list for **Operator Class**.

**Note:** Two operator classes in the same schema can have the same name only if they are for different index methods.

The **Comment** tab is supported from PostgreSQL 8.0 or later.

### General Properties

#### Owner

The owner of the operator class function.

**Note:** Support from PostgreSQL 8.0 or later.

### **Schema of Data Type and Data Type**

The column data type that this operator class is for.

### **Index method**

The name of the index method this operator class is for.

### **Schema of Storage Type and Storage type**

The data type actually stored in the index. Normally this is the same as the column data type, but some index methods (*GIN* and *GiST* for now) allow it to be different. The *STORAGE* clause must be omitted unless the index method allows a different type to be used.

### **Operator family**

The name of the existing operator family to add this operator class to. If not specified, a family named the same as the operator class is used (creating it, if it doesn't already exist).

**Note:** Support from PostgreSQL 8.3 or later.

### **Default operator class**

With this option selected, the operator class will become the default operator class for its data type. At most one operator class can be the default for a specific data type and index method.

## **Operators**

### **Strategy number**

The index method's strategy number for an operator associated with the operator class.

### **Schema of operator and Operator name**

The operator associated with the operator class.

### **Recheck**

With this option selected, the index is "lossy" for this operator, and so the rows retrieved using the index must be rechecked to verify that they actually satisfy the qualification clause involving this operator.

**Note:** Before PostgreSQL 8.4, the *OPERATOR* clause could include a *RECHECK* option. This is no longer supported because whether an index operator is "lossy" is now determined on-the-fly at runtime. This allows efficient handling of cases where an operator might or might not be lossy.

## **Functions**


### **Support number**

The index method's support procedure number for a function associated with the operator class.

### **Schema of function and Function name**

The function that is an index method support procedure for the operator class.

## PostgreSQL Sequences

Sequence involves creating and initializing a new special single-row table. It is usually used to generate unique identifiers for rows of a table. Click  -> **Sequence** to open an object list for **Sequence**.

### General Properties

#### Owner

The owner of the sequence function.

**Note:** Support from PostgreSQL 8.0 or later.

#### Increment

Specify which value is added to the current sequence value to create a new value. A positive value will make an ascending sequence, a negative one a descending sequence. The default value is 1.

#### Current value

The starting value of the sequence.

#### Minimum

Determine the minimum value a sequence can generate.

#### Maximum

Determine the maximum value for the sequence.

#### Cache

Specify how many sequence numbers are to be preallocated and stored in memory for faster access.

#### Cycled


This option allows the sequence to wrap around when the maxvalue or minvalue has been reached by an ascending or descending sequence respectively. If the limit is reached, the next number generated will be the minvalue maxvalue, respectively. Otherwise, any calls to nextval after the sequence has reached its maximum value will return an error.

#### Add owned by

Choose the **Owned by table** and **Owned by column** so that the sequence is associated with a specific table column, such that if that column (or its whole table) is dropped, the sequence will be automatically dropped as well. The specified table must have the same owner and be in the same schema as the sequence.

**Note:** Support from PostgreSQL 8.2 or later.

## PostgreSQL Triggers

Triggers are database operations that are automatically performed when a specified database event occurs. Click  -> **Trigger** to open an object list for **Trigger**.

See [Triggers](#) for details.

## General Properties

### Constraint

Check this box to create a constraint trigger.

### Trigger Type

Choose the type of the trigger: Table or View.

**Note:** Support from PostgreSQL 9.0 or later.

### Table name or View name

Choose a table or view.

### BEFORE

The trigger can be specified to fire before the operation is attempted on a row.

### AFTER

The trigger can be specified to fire after the operation is attempted on a row.

### INSTEAD OF

The trigger can be specified to fire instead of the operation is attempted on a row.

### INSERT/UPDATE/DELETE

Choose the events that activate the trigger.

#### Insert

The trigger is activated whenever a new row is inserted.

#### Update

The trigger is activated whenever a row is modified.

#### Delete

The trigger is activated whenever a row is deleted.

### TRUNCATE

Trigger defined to fire for TRUNCATE.

### Update Of Fields

Specify a list of columns. The trigger will only fire if at least one of the listed columns is mentioned as a target of the UPDATE command.

### STATEMENT

Specify the trigger procedure should be fired once per SQL statement.

## ROW

Specify the trigger procedure should be fired once for every row affected by the trigger event.

## When

Specify a Boolean WHEN condition, which will be tested to see whether the trigger should be fired.

**Note:** Support from PostgreSQL 9.0 or later.

## Function Schema and Function name

A user-supplied function that is declared as taking no arguments and returning type trigger, which is executed when the trigger fires.

## Function Arguments

An optional comma-separated list of arguments to be provided to the function when the trigger is executed. The arguments are literal string constants. Simple names and numeric constants may be written here, too, but they will all be converted to strings. Please check the description of the implementation language of the trigger function about how the trigger arguments are accessible within the function; it may be different from normal function arguments.

## Constraint

### Deferrable

The constraint can be deferred.

### INITIALLY IMMEDIATE

The constraint is checked after each statement.

### INITIALLY DEFERRED


The constraint is checked only at the end of the transaction.

## Referenced Table Schema and Referenced Table Name

The schema and the name of another table referenced by the constraint.

## PostgreSQL Trigger Functions

Trigger Function can be created with PL/pgSQL and referenced within a PostgreSQL trigger definition. The term "trigger function" is simply a way of referring to a function that is intended to be invoked by a trigger. Triggers define operations that are performed when a specific event occurs within the database. A PL/pgSQL trigger function can be referenced by a trigger as the operation to be performed when the trigger's event occurs.

The definition of a trigger and the definition of its associated trigger function are two different things. A trigger is defined with the SQL *CREATE TRIGGER* command, whereas trigger functions are defined using the SQL *CREATE FUNCTION* command. Click  -> **Trigger Function** to open an object list for **Trigger Function**.

See [Triggers](#) for details.

## Definition

Definition consists of a valid SQL procedure statement. This can be a simple statement such as *SELECT* or *INSERT*, or it can be a compound statement written using *BEGIN* and *END*. Compound statements can contain declarations, loops, and other control structure statements.

## Parameter

Define trigger function parameter.

## Return type schema and Return Type

The return type of the trigger function.

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

## Advanced Properties

### Owner

The owner of the trigger function.

**Note:** Support from PostgreSQL 8.0 or later.

### Language

The name of the language that the function is implemented in. May be C, internal, or the name of a user-defined procedural language. For backward compatibility, the name may be enclosed by single quotes.

### Volatility

These attributes inform the query optimizer about the behavior of the function. At most one choice may be specified. If none of these appear, VOLATILE is the default assumption.

IMMUTABLE	The function cannot modify the database and always returns the same result when given the same argument values; that is, it does not do database lookups or otherwise use information not directly present in its argument list. If this option is given, any call of the function with all-constant arguments can be immediately replaced with the function value.
STABLE	The function cannot modify the database, and that within a single table scan it will consistently return the same result for the same argument values, but that its result could change across SQL statements. This is the appropriate selection for functions whose results depend on database lookups, parameter variables (such as the current time zone), etc. Also note that the <code>current_timestamp</code> family of functions qualify as stable, since their values do not change within a transaction.
VOLATILE	The function value can change even within a single table scan, so no optimizations can be made. Relatively few database functions are volatile in this sense; some examples are <code>random()</code> , <code>curval()</code> , <code>timeofday()</code> . But note that any function that has side-effects must be classified volatile, even if its result is quite predictable, to prevent calls from being optimized away; an example is <code>setval()</code> .



### Security of definer

Specify that the function is to be executed with the privileges of the user that created it.

### Returns Set

Indicate that the function will return a set of items, rather than a single item.

### Strict

Indicate that the function always returns null whenever any of its arguments are null. If this parameter is specified, the function is not executed when there are null arguments; instead a null result is assumed automatically.

### Estimated cost

A positive number giving the estimated execution cost for the function, in units of `cpu_operator_cost`. If the function returns a set, this is the cost per returned row. If the cost is not specified, 1 unit is assumed for C-language and internal functions, and 100 units for functions in all other languages. Larger values cause the planner to try to avoid evaluating the function more often than necessary.

**Note:** Support from PostgreSQL 8.2 or later.

### Estimated rows

A positive number giving the estimated number of rows that the planner should expect the function to return. This is only allowed when the function is declared to return a set.

**Note:** Support from PostgreSQL 8.2 or later.

### Configuration parameter


The specified configuration parameter to be set to the specified value when the function is entered, and then restored to its prior value when the function exits.

**Note:** Support from PostgreSQL 8.2 or later.

### Debug (Available only in Full Version)

To debug the PL/pgSQL function, click  **Debug Function** on the toolbar to launch the [PostgreSQL Debugger](#).

## PostgreSQL Types

Type registers a new data type for use in the current database. If a schema name is given then the type is created in the specified schema. Otherwise it is created in the current schema. The type name must be distinct from the name of any existing type or domain in the same schema. (Because tables have associated data types, the type name must also be distinct from the name of any existing table in the same schema.) Click  -> **Type** to open an object list for **Type**.

**Base types** are those, like `int4`, that are implemented below the level of the SQL language (typically in a low-level language such as C). They generally correspond to what are often known as abstract data types. PostgreSQL can only operate on such types through functions provided by the user and only understands the behavior of such types to the

extent that the user describes them. Base types are further subdivided into scalar and array types. For each scalar type, a corresponding array type is automatically created that can hold variable-size arrays of that scalar type.

**Composite types**, or row types, are created whenever the user creates a table; it's also possible to define a "stand-alone" composite type with no associated table. A composite type is simply a list of base types with associated field names. A value of a composite type is a row or record of field values. The user can access the component fields from SQL queries.

**Enumerated (Enum) types** are data types that are comprised of a static, predefined set of values with a specific order. They are equivalent to the enum types in a number of programming languages. An example of an enum type might be the days of the week, or a set of status values for a piece of data.

**Range types** are data types representing a range of values of some element type (called the range's subtype).

**Note:** Enum Type was added in PostgreSQL 8.3. Range Type was added in PostgreSQL 9.2.

## General Properties for Base Type

### Input Schema and Input

The function that converts data from the type's external textual form to its internal form.

### Output Schema and Output

The function that converts data from the type's internal form to its external textual form.

### Length

A numeric constant that specifies the length in bytes of the new type's internal representation. The default assumption is that it is variable-length.

### Variable

Check this option if the type length is unknown.

### Default

The default value for the data type. If this is omitted, the default is null.

### Element

The type being created is an array; this specifies the type of the array elements.

### Delimiter

The delimiter character to be used between values in arrays made of this type.

### Alignment

The storage alignment requirement of the data type. If specified, it must be char, int2, int4, or double; the default is int4.

### Storage

The storage strategy for the data type. If specified, must be plain, external, extended, or main; the default is plain.

### Pass by value

Indicate that values of this data type are passed by value rather than by reference.

### Owner

The owner of the type.

**Note:** Support from PostgreSQL 8.0 or later.

### Advanced Properties for Base Type

The **Advanced** tab is supported from PostgreSQL 7.4 or later.

### Receive Schema and Receive

The function that converts data from the type's external binary form to its internal form.

### Send Schema and Send

The function that converts data from the type's internal form to its external binary form.

### Analyze Schema and Analyze

The function that performs statistical analysis for the data type.

**Note:** Support from PostgreSQL 8.0 or later.

### Type Modifier Input Schema and Type Modifier Input

The function that converts an array of modifier(s) for the type into internal form.

**Note:** Support from PostgreSQL 8.3 or later.

### Type Modifier Output Schema and Type Modifier Output

The function that converts the internal form of the type's modifier(s) to external textual form.

**Note:** Support from PostgreSQL 8.3 or later.

### General Properties for Composite Type

#### Name

The name of an attribute (column) for the composite type.

#### Type

The name of an existing data type to become a column of the composite type.

#### Length and Scale

Use the **Length** edit box to define the length of the field and use **Scale** edit box to define the number of digits after the decimal point. (if required for the selected data type)

## Dimensions

The dimensions of array specifiers.

## Collate

The name of an existing collation to be associated with a column of a composite type.

## Owner

The owner of the type.

**Note:** Support from PostgreSQL 8.0 or later.

## General Properties for Enum Type

### Label

A string literal representing the textual label associated with one value of an enum type.

### Owner

The owner of the type.

## General Properties for Range Type

### Subtype Schema and Subtype

The schema and the name of the element type that the range type will represent ranges of.

### Operator Class Schema and Operator Class

The schema and the name of a b-tree operator class for the subtype.

### Collate

The name of an existing collation to be associated with a column with a range type.

### Canonical Schema and Canonical

The schema and the name of the canonicalization function for the range type.


### Subtype Diff Schema and Subtype Diff

The schema and the name of a difference function for the subtype.

### Owner

The owner of the type.

## PostgreSQL Tablespaces

A tablespace allows superusers to define an alternative location on the file system where the data files containing database objects (such as tables and indexes) may reside. Click  -> **Tablespace** to open an object list for **Tablespace**.

**Note:** Tablespace was added in PostgreSQL 8.0.

The **Comment** tab is supported from PostgreSQL 8.2 or later.

## General Properties


### Location

The directory that will be used for the tablespace. The directory must be empty and must be owned by the PostgreSQL system user. The directory must be specified by an absolute path name.

### Owner

The name of the user who will own the tablespace. If omitted, defaults to the user executing the command. Only superusers may create tablespaces, but they can assign ownership of tablespaces to non-superusers.

## PostgreSQL Casts

A cast specifies how to perform a conversion between two data types. Click  -> **Cast** to open an object list for **Cast**.

**Note:** The **Comment** tab is supported from PostgreSQL 8.0 or later.

## General Properties

### Schema of source type and Source type

The schema and name of the source data type of the cast.

### Schema of target type and Target type

The schema and name of the target data type of the cast.

### Schema of function and Function

The function used to perform the cast. The function name may be schema-qualified. If it is not, the function will be looked up in the schema search path. The function's result data type must match the target type of the cast.

If no function is specify, indicates that the source type and the target type are binary compatible, so no function is required to perform the cast.


### Implicit

Indicate that the cast may be invoked implicitly in any context.

### Assignment

Indicate that the cast can be invoked implicitly in assignment contexts.

## PostgreSQL Foreign Servers

A foreign server typically encapsulates connection information that a foreign-data wrapper uses to access an external data resource. Additional user-specific connection information may be specified by means of user mappings. Click  -> **Foreign Server** to open an object list for **Foreign Server**.

**Note:** Support from PostgreSQL 8.4 or later.

To install the `postgres_fdw` extension for accessing data stored in external PostgreSQL servers, you can right-click anywhere the Object List pane and select **Install postgres\_fdw Extension**.

## General Properties

### FDW Name

The name of the foreign-data wrapper that manages the server.

### Options

Define the connection details of the server, but the actual names and values are dependent on the server's foreign-data wrapper.

### Server Type

Specify the server type.

### Server Version

Specify the server version.

### Owner

The owner of the foreign server.

## User Mappings


### Name

The name of an existing user that is mapped to foreign server.

### Options

Specify the options of the user mapping. The options typically define the actual user name and password of the mapping.

## PostgreSQL Languages

Language can register a new procedural language with a PostgreSQL database. Subsequently, functions and trigger procedures can be defined in this new language. The user must have the PostgreSQL superuser privilege to register a new language. Click  -> **Language** to open an object list for **Language**.

**Note:** The **Comment** tab is supported from PostgreSQL 8.0 or later.

## General Properties

### Owner

The owner of the language.

**Note:** Support from PostgreSQL 8.3 or later.

### Schema of handler and Handler

Call Handler is the name of a previously registered function that will be called to execute the procedural language functions. The call handler for a procedural language must be written in a compiled language such as C with version 1 call convention and registered with PostgreSQL as a function taking no arguments and returning the *language\_handler* type, a placeholder type that is simply used to identify the function as a call handler.

### Schema of validator and Validator

Validator function is the name of a previously registered function that will be called when a new function in the language is created, to validate the new function. If no validator function is specified, then a new function will not be checked when it is created. The validator function must take one argument of type oid, which will be the OID of the to-be-created function, and will typically return void.

A validator function would typically inspect the function body for syntactical correctness, but it can also look at other properties of the function, for example if the language cannot handle certain argument types. To signal an error, the validator function should use the ereport() function. The return value of the function is ignored.

### Trusted

Specify that the call handler for the language is safe, that is, it does not offer an unprivileged user any functionality to bypass access restrictions. If this key word is omitted when registering the language, only users with the PostgreSQL superuser privilege can use this language to create new functions.

## SQLite Objects

To start working with the server objects, you should create and open a connection.

You can attach a database, right-click the opened connection and choose **Attach Database** and enter the following information.


Option	Description
Database File	Set the file path for a database.
Database Name	Enter the database name which displays in Navicat.
Encrypted	Enable this option and provide <b>Password</b> when connecting to an encrypted SQLite database.



To detach a database, right-click it in the Connection pane and choose **Detach Database**.

If you want to encrypt or decrypt a database, simply right-click it in the Connection pane and choose **Encrypt Database** or **Decrypt Database**.

A special table named sqlite\_master stores the complete database schema. To view the sqlite\_master table, right-click the database and select **View Master Table** from the pop-up menu.

## SQLite Tables







Relational databases use tables to store data. All operations on data are done on the tables themselves or produce another table as the result. A table is a set of rows and columns, and their intersections are fields. From a general perspective, columns within a table describe the name and type of data that will be found by row for that column's fields. Rows within a table represent records composed of fields that are described from left to right by their corresponding column's name and type. Each field in a row is implicitly correlated with each other field in that row. Click  to open an object list for **Table**.

When open a table with graphical fields, right-click a table and select the  **Open Table (Quick)** from the pop-up menu. Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell. If you do wish Navicat loads all your images while opening the table, use  **Open Table**.

To empty a table, right-click the selected table and choose **Empty Table** from the pop-up menu.

## SQLite Table Fields

In the **Fields** tab, just simply click a field for editing. A right-click displays the pop-up menu or using field toolbar, you can create new, insert, move and drop the selected field. To search a field name, choose **Edit -> Find** or press CTRL+F.

Button	Description
 Add Field	Add a field to the table.
 Insert Field	Insert a field above an existing field.
 Delete Field	Delete the selected field.
 Primary Key	Set the field as a Primary Key.
 Move Up/  Move Down	Move the selected field up/down.

Use the **Name** edit box to set the field name. Note that the name of the field must be unique among all the field names in the table.

The **Type** drop-down list defines the type (storage class) of the field data. See [SQLite 2 Data Types](#) and [SQLite 3 Data Types](#) for details.

Use the **Length** edit box to define the length of the field and use **Decimals** edit box to define the number of digits after the decimal point (the scale).

### Not null

Check this box to not allow the NULL values for the field.

### Primary Key

A Primary Key is a single field or combination of fields that uniquely defines a record. None of the fields that are part of the primary key can contain a null value.



## Field's Properties

**Note:** The following options depend on the field type you are chosen.

### Default

To set the default value for the field.

### Collation

To specify the text collating function to use when comparing text entries for the column.

BINARY	Compare string data using memcmp(), regardless of text encoding.
NOCASE	The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.
RTRIM	The same as binary, except that trailing space characters are ignored.

**Note:** Support in SQLite 3.

### Not null ON CONFLICT

To specify an algorithm used to resolve constraint conflicts if **Not null** option is checked.

ROLLBACK	When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.
ABORT	When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.
FAIL	When a constraint violation occurs, the command aborts with a return code SQLITE_CONSTRAINT. But any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.
IGNORE	When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned when the IGNORE conflict resolution algorithm is used.
REPLACE	When a UNIQUE constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally following REPLACE. No error is returned by the REPLACE conflict resolution. If a NOT NULL constraint violation occurs, the NULL value is replaced by the default value for that column. If the column has no

	default value, then the ABORT algorithm is used. If a CHECK constraint violation occurs then the IGNORE algorithm is used.
--	----------------------------------------------------------------------------------------------------------------------------



### Auto Increment

The AUTO INCREMENT attribute can be used to generate a unique identity for new rows. To start with the AUTO INCREMENT value other than 1, you can set that value in Options tab.


## SQLite Table Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table.

In the **Indexes** tab, just simply click an index field for editing. By using the index toolbar, you can create new, edit and delete the selected index field.

Button	Description
 Add Index	Add an index to the table.
 Delete Index	Delete the selected index.

Use the **Name** edit box to set the index name.

To include field(s) in the index, just simply double-click the **Fields** or click  to open the editor for editing.

### Unique

All values of the indexed column(s) must only occur once.

### Fields Editor

Select the field(s) from the **Name** list. You can also use the arrow buttons to change the index field(s) order.

### Collation

To define a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column.

BINARY	Compares string data using memcmp(), regardless of text encoding.
NOCASE	The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.
RTRIM	The same as binary, except that trailing space characters are ignored.

**Note:** Support in SQLite 3.



### Sort Order

To indicate sort order - ascending "ASC" or descending "DESC".

## SQLite Table Foreign Keys


A foreign key is a field in a relational table that matches the primary key column of another table.

In the **Foreign Keys** tab, just simply click a foreign key field for editing. By using the foreign key toolbar, you can create new, edit and delete the selected foreign key field.

Button	Description
 Add Foreign Key	Add a foreign key to the table.
 Delete Foreign Key	Delete the selected foreign key.

Use the **Name** edit box to enter a name for the new key.

Use the **Referenced Table** drop-down list to select a foreign table.

To include field(s)/referenced field(s) to the key, just simply double-click the **Fields/Referenced Fields** field or click  to open the editor(s) for editing.

The **On Delete** and **On Update** drop-down list define the type of the actions to be taken.

RESTRICT	The "RESTRICT" action means that the application is prohibited from deleting (for ON DELETE RESTRICT) or modifying (for ON UPDATE RESTRICT) a parent key when there exists one or more child keys mapped to it.
NO ACTION	Configuring "NO ACTION" means just that: when a parent key is modified or deleted from the database, no special action is taken.
CASCADE	A "CASCADE" action propagates the delete or update operation on the parent key to each dependent child key. For an "ON DELETE CASCADE" action, this means that each row in the child table that was associated with the deleted parent row is also deleted. For an "ON UPDATE CASCADE" action, it means that the values stored in each dependent child key are modified to match the new parent key values.
SET NULL	If the configured action is "SET NULL", then when a parent key is deleted (for ON DELETE SET NULL) or modified (for ON UPDATE SET NULL), the child key columns of all rows in the child table that mapped to the parent key are set to contain SQL NULL values.
SET DEFAULT	The "SET DEFAULT" actions are similar to "SET NULL", except that each of the child key columns is set to contain the columns default value instead of NULL.

### Deferred

Deferred foreign key constraints are not checked until the transaction tries to COMMIT.



Related topic:

[Foreign Keys Data Selection](#)

## SQLite Table Uniques

Unique constraints ensure that the data contained in a column or a group of columns is unique with respect to all the rows in the table.

In the **Uniques** tab, just simply click an unique field for editing. By using the unique toolbar, you can create new, edit and delete the selected unique field.

Button	Description
 Add Unique	Add an unique to the table.
 Delete Unique	Delete the selected unique.

Use the **Name** edit box to set the unique name.

To set field(s) as unique, just simply double-click the **Fields** field or click  to open the editor(s) for editing.

### ON CONFLICT

To specify an algorithm used to resolve constraint conflicts.

ROLLBACK	When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.
ABORT	When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.
FAIL	When a constraint violation occurs, the command aborts with a return code SQLITE_CONSTRAINT. But any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.
IGNORE	When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned when the IGNORE conflict resolution algorithm is used.
REPLACE	When a UNIQUE constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally following REPLACE. No error is returned by the REPLACE conflict resolution. If a NOT NULL constraint violation occurs, the NULL value is replaced by the default value for that column. If the column has no default value, then the ABORT algorithm is used. If a CHECK constraint violation occurs then

	the IGNORE algorithm is used.
--	-------------------------------

## Fields Editor

Select the field(s) from the **Name** list. To remove the fields from the unique, uncheck them in the same way. You can also use the arrow buttons to change the unique field(s) order.

## Collation

To define a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column.

BINARY	Compares string data using memcmp(), regardless of text encoding.
NOCASE	The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded. SQLite does not attempt to do full UTF case folding due to the size of the tables required.
RTRIM	The same as binary, except that trailing space characters are ignored.

**Note:** Support in SQLite 3.

## Sort Order



To indicate sort order - ascending "ASC" or descending "DESC".

## SQLite Table Checks

A check constraint allows you to specify that the value in a certain column must satisfy a Boolean (truth-value) expression.

In the **Checks** tab, just simply click a check field for editing. By using the check toolbar, you can create new, edit and delete the selected check field.

**Note:** Checks are supported from SQLite version 3.3.0 or later.

Button	Description
 Add Check	Add a check to the table.
 Delete Check	Delete the selected check.

Use the **Name** edit box to set the check name.

## Check

Set the condition for checking, e.g. "field\_name1 > 0 AND field\_name2 > field\_name1" in the **Check** edit box.



## Definition

Type in the definition for the check constraint.

## SQLite Table Triggers

A trigger is a database operation that is automatically performed when a specified database event occurs.

In the **Triggers** tab, just simply click a trigger field for editing. By using the trigger toolbar, you can create new, edit and delete the selected trigger field.

Button	Description
 Add Trigger	Add a trigger to the table.
 Delete Trigger	Delete the selected trigger.

### Name

Set the trigger name.

### Fires

Determine when the trigger actions will be executed relative to the insertion, modification or removal of the associated row.

### Insert

Fire the trigger whenever an INSERT statement adds a row to a table.

### Update

Fire the trigger whenever an UPDATE statement changes a value in one of the columns specified in **Update Of Fields**. If no **Update Of Fields** are present, the trigger will be fired whenever an UPDATE statement changes a value in any column of the table.

### Delete

Fire the trigger whenever a DELETE statement removes a row from the table.

### Update Of Fields

Specify the fields for UPDATE statement trigger upon necessary.

### Definition

Type in the definition for the trigger.

### When Clause

Specify the trigger condition, which is a SQL condition that must be satisfied for the database to fire the trigger.

## SQLite Table Options

### Primary Key ON CONFLICT

To specify an algorithm used to resolve primary key constraint conflicts.

ROLLBACK	When a constraint violation occurs, an immediate ROLLBACK occurs, thus ending the
----------	-----------------------------------------------------------------------------------

	current transaction, and the command aborts with a return code of SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm works the same as ABORT.
ABORT	When a constraint violation occurs, the command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed so changes from prior commands within the same transaction are preserved. This is the default behavior.
FAIL	When a constraint violation occurs, the command aborts with a return code SQLITE_CONSTRAINT. But any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.
IGNORE	When a constraint violation occurs, the one row that contains the constraint violation is not inserted or changed. But the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned when the IGNORE conflict resolution algorithm is used.
REPLACE	When a UNIQUE constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally following REPLACE. No error is returned by the REPLACE conflict resolution. If a NOT NULL constraint violation occurs, the NULL value is replaced by the default value for that column. If the column has no default value, then the ABORT algorithm is used. If a CHECK constraint violation occurs then the IGNORE algorithm is used.

### Auto Increment


Set/Reset the **Auto Increment** value in the edit field. The auto increment value indicates the value for next record.




### WITHOUT ROWID

Omit the rowid (and "oid" and "\_rowid\_").

**Note:** Support from SQLite 3.8.2 or later.

## SQLite Views

Views are useful for allowing users to access a set of tables as if it were a single table, and limiting their access to just that. Views can also be used to restrict access to rows (a subset of a particular table). Click  to open an object list for **View**.

Button	Description
 Preview	Preview the result of the view.
 Explain	Show the Query Plan of the view.
 Beautify SQL	Format the codes with the Beautify SQL settings in Editor.

**Note:** You can choose to show the Result tab below the editor or in a new tab by selecting **View -> Show Result -> Below Query Editor** or **In a New Tab**.

## View Builder (Available only in Full Version)

**View Builder** allows you to build views visually. It allows you to create and edit views without knowledge of SQL. See [Query Builder](#) for details.

## Definition


You can edit the view definition as SQL statement (SELECT statement it implements).

**Hint:** To customize the view of the editor and find out more features for sql editing, see [Editor Advanced Features](#).

## View Viewer

**View Viewer** displays the view data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See [Table Viewer](#) for details.

# SQLite Indexes

Index provides a faster access path to table data. It is created using one or more columns of a table to speed SQL statement execution on that table. Click  to open an object list for **Index**.

## General Properties

### Type

The types of the index.

Normal	A normal index does not impose restrictions on the column values.
Unique	An unique index indicates that no two rows of a table have duplicate values in the key columns.

### Table name

The table that contains the index.

### Name

To define the field.

### Collate

To define a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column.

BINARY	Compare string data using memcmp(), regardless of text encoding.
NOCASE	The same as binary, except the 26 upper case characters of ASCII are folded to their lower case equivalents before the comparison is performed. Note that only ASCII characters are case folded.




	SQLite does not attempt to do full UTF case folding due to the size of the tables required.
RTRIM	The same as binary, except that trailing space characters are ignored.

**Note:** Support in SQLite 3.

### Sort Order

To indicate sort order - ascending "ASC" or descending "DESC".

## SQLite Triggers

Triggers are database operations that are automatically performed when a specified database event occurs. Click  to open an object list for **Trigger**.

See [Triggers](#) for details.

### General Properties

#### Trigger Type

Define the trigger type: TABLE or VIEW.

#### Table name or View name

Choose a table or view.

#### BEFORE

The trigger can be specified to fire before the operation is attempted on a row.

#### AFTER

The trigger can be specified to fire after the operation is attempted on a row.

#### INSTEAD OF

The trigger can be specified to fire instead of the operation is attempted on a row.

#### When

Specify the trigger condition for the database to fire the trigger.

#### Insert

The trigger is activated whenever adding a row to a table.

#### Delete

The trigger is activated whenever removing a row from the table.

#### Update

The trigger is activated whenever changing a value in one of the fields selected in **Update Of Fields**.

## Update Of Fields



Specify the fields for UPDATE statement trigger upon necessary.

## Definition

You can edit valid SQL statements in the trigger definition inside *BEGIN* and *END*.

# SQL Preview

The **SQL Preview** tab shows the CREATE statement and necessary SQL statements of the database or schema object.

For some database or schema objects, you can use the below drop-down list to show the SQL which will be run when pressing  **Save** or  **Save As** button.

# Maintain

Navicat provides a complete solution for maintaining databases and their database objects in MySQL, Oracle, PostgreSQL, SQLite and MariaDB.

To maintain server objects, you can right-click it and select **Maintain** from the pop-up menu.

## Maintain MySQL/MariaDB

### Maintain Table

#### Analyze Tables

Analyze and store the key distribution for the table. During the analysis, the table is locked with a read lock for MyISAM and BDB. For InnoDB the table is locked with a write lock. Currently, MySQL supports analyzing only for MyISAM, BDB, and InnoDB tables.

#### Check Tables

Check a table or tables for errors. Currently, MySQL supports checking only for MyISAM, InnoDB and ARCHIVE tables. For MyISAM tables, the key statistics are updated as well.

Normal	Run the CHECK TABLE statement without an extra option.
Quick	Don't scan the rows to check for wrong links.
Fast	Check only tables that have not been closed properly.
Changed	Only check tables which have been changed since last check or haven't been closed properly.
Extended	Do a full key lookup for all keys for each row. This ensures that the table is 100 % consistent, but will take a long time.

## Optimize Tables

The main reason for optimizing your table is to reclaim unused space and to defragment the data file. You should optimize a table if you have deleted a large part of a table or if you have made many changes to a table with variable-length rows (tables that have VARCHAR, BLOB, or TEXT columns). Deleted records are maintained in a linked list and subsequent INSERT operations reuse old row positions. Currently, MySQL supports optimizing only for MyISAM, InnoDB and BDB tables.

## Repair Tables

Repair a possibly corrupted table and returns a result set.

Quick	Repair Table tries to repair only the index tree.
Extended	MySQL creates the index row by row instead of creating one index at a time with sorting.

## Get Rows Count

Count the number of rows in the table.

## Maintain Oracle

### Maintain Table

#### Enable Table Lock

Choose Enable Table Lock to enable table locks, thereby allowing DDL operations on the table. All currently executing transactions must commit or roll back before Oracle Database enables the table lock.

#### Disable Table Lock

Choose Disable Table Lock to disable table locks, thereby preventing DDL operations on the table.

#### Enable Row Movement

Choose Enable Row Movement to allow the database to move a row, thus changing the rowid.

#### Disable Row Movement

Choose Disable Row Movement if you want to prevent the database from moving a row, thus preventing a change of rowid.

#### Shrink Space

Shrink Space is to compact the table segment. This clause is valid only for segments in tablespaces with automatic segment management. By default, Oracle database compacts the segment, adjusts the high water mark, and releases the recuperated space immediately.

Compacting the segment requires row movement. Therefore, you must enable row movement for the table you want to shrink before shrink space. Further, if your application has any rowid-based triggers, you should disable them before issuing this clause.

## **Move**

Move relocates data of a nonpartitioned table or of a partition of a partitioned table into a new segment, optionally in a different tablespace, and optionally modify any of its storage attributes.

## **Collect Statistics**

Collect Statistics analyzes the contents of tables. When you analyze a table, the database collects statistics about expressions occurring in any function-based indexes as well. Therefore, be sure to create function-based indexes on the table before analyzing the table.

## **Validate Structure**

Validate Structure verifies the integrity of the structure of a table. The statistics collected by this clause are not used by the Oracle database optimizer. If the structure is valid, no error is returned. However, if the structure is corrupt, an error message will be shown.

For a table, Oracle database verifies the integrity of each of the data blocks and rows.

## **Maintain View**

### **Compile**

To recompile the view specification or body.

## **Maintain Function/Procedure**

### **Compile**

To recompile the function/procedure specification or body.

### **Compile for Debug**

To recompile the function/procedure specification or body and instruct the PL/SQL compiler to generate and store the code for use by the PL/SQL debugger.

## **Maintain Index**

### **Rebuild**

To re-create an existing index or one of its partitions or subpartitions. If the index is marked unusable, then a successful rebuild will mark it usable.

### **Make Unusable**

To make the index unusable. An unusable index must be rebuilt, or dropped and re-created, before it can be used.

### **Coalesce**

To instruct Oracle database to merge the contents of index blocks where possible to free blocks for reuse.

### **Compute Statistics**

To compute the statistics of the index.

### **Monitoring Usage**

To begin monitoring the index. Oracle Database first clears existing information on index use, and then monitors the index for use until choosing No Monitoring Usage.

### **No Monitoring Usage**

To terminate monitoring of the index.

## **Maintain Java**

### **Compile or Resolve**

To resolve the primary Java class schema object.

### **Set AuthID Current User**

Set the invoker rights to AUTHID CURRENT\_USER.

### **Set AuthID Definer**

Set the invoker rights to AUTHID DEFINER.

## **Maintain Materialized View**

### **Enable Row Movement**

To enable row movement.

### **Shrink**

To compact the materialized view segment. By default, Oracle database compacts the segment, adjusts the high water mark, and releases the recuperated space immediately.

### **Compile**

To explicitly revalidate a materialized view. If an object upon which the materialized view depends is dropped or altered, then the materialized view remains accessible, but it is invalid for query rewrite. You can choose this option to explicitly revalidate the materialized view to make it eligible for query rewrite.

### **Force Refresh**

To perform a refresh.

## **Maintain Materialized View Log**

### **Enable Row Movement**

To enable row movement. Row movement indicates that rowids will change after the flashback occurs.

### **Disable Row Movement**

To disable row movement.

### **Shrink Space**

To compact the materialized view log segments. By default, Oracle database compacts the segment, adjusts the high water mark, and releases the recuperated space immediately.

## Maintain Package

### Compile

To recompile the package specification or body.

### Compile Debug

To recompile the package specification or body and instruct the PL/SQL compiler to generate and store the code for use by the PL/SQL debugger.

## Maintain Trigger

### Enable

To enable the trigger.

### Disable

To disable the trigger.

### Compile

To explicitly compile the trigger, whether it is valid or invalid. Explicit recompilation eliminates the need for implicit run-time recompilation and prevents associated run-time compilation errors and performance overhead.

### Compile for Debug

To recompile the trigger and instruct the PL/SQL compiler to generate and store the code for use by the PL/SQL debugger.

## Maintain Type

### Compile

To compile the type specification and body.

### Compile Debug

To recompile the type specification or body and instruct the PL/SQL compiler to generate and store the code for use by the PL/SQL debugger.

## Maintain XML Schema

### Compile

To re-compile an already registered XML schema. This is useful for bringing a schema in an invalid state to a valid state.

### Purge

To remove the XML Schema completely from Oracle XML DB in Oracle 11g.

## Maintain Tablespace

### Read Only

To place the tablespace in transition read-only mode. In this state, existing transactions can complete (commit or roll back), but no further DML operations are allowed to the tablespace except for rollback of existing transactions that previously modified blocks in the tablespace.

### Read Write

To indicate that write operations are allowed on a previously read-only tablespace.

### Online

To take the tablespace online.

### Offline

To take the tablespace offline.

Normal	To flush all blocks in all datafiles in the tablespace out of the system global area (SGA).
Temporary	Oracle database performs a checkpoint for all online datafiles in the tablespace but does not ensure that all files can be written.
Immediate	Oracle database does not ensure that tablespace files are available and does not perform a checkpoint.

### Coalesce

To combine all contiguous free extents into larger contiguous extents for each datafile in the tablespace.

### Shrink Space

To reduce the amount of space the tablespace is taking. This is valid only for temporary tablespaces in Oracle 11g.

## Maintain User

### Expire Password

To set password of user account will expire.

### Lock Account

To lock user account.

### Unlock Account

To unlock user account.

## Maintain PostgreSQL

### Maintain Database, Table and Materialized View

#### Analyze Database, Analyze Tables and Analyze Materialized Views

Collect statistics about the contents of tables in the database, and stores the results in the system table *pg\_statistic*. Subsequently, the query planner uses these statistics to help determine the most efficient execution plans for queries. **Analyze Database** examines every table in the current database.

When VERBOSE is specified, ANALYZE emits progress messages to indicate which table is currently being processed. Various statistics about the tables are printed as well. It is enabled in Navicat by default.

### **Vacuum Database, Vacuum Tables and Vacuum Materialized Views**

Reclaim storage occupied by deleted tuples. In normal PostgreSQL operation, tuples that are deleted or obsoleted by an update are not physically removed from their table; they remain present until a Vacuum is done. Therefore it's necessary to do Vacuum periodically, especially on frequently-updated tables. **Vacuum Database** examines every table in the current database.

When VERBOSE is specified, VACUUM emits progress messages to indicate which table is currently being processed. Various statistics about the tables are printed as well. It is enabled in Navicat by default.

Vacuum	Run the VACUUM VERBOSE statement without extra options.
Vacuum Analyze	Update statistics used by the planner to determine the most efficient way to execute a query.
Vacuum Full	Select "full" vacuum, which may reclaim more space, but takes much longer and exclusively locks the table.
Vacuum Full Analyze	Select "full" vacuum, which may reclaim more space, but takes much longer and exclusively locks the table. Update statistics used by the planner to determine the most efficient way to execute a query.
Vacuum Freeze	Select aggressive "freezing" of tuples.
Vacuum Freeze Analyze	Select aggressive "freezing" of tuples. Update statistics used by the planner to determine the most efficient way to execute a query.

### **Reindex Database, Reindex Tables and Reindex Materialized Views**

Rebuild an index using the data stored in the index's table, replacing the old copy of the index. There are several scenarios in which to use Reindex:

- An index has become corrupted, and no longer contains valid data.
- An index has become "bloated", that it contains many empty or nearly-empty pages.
- You have altered a storage parameter (such as fill factor) for an index, and wish to ensure that the change has taken full effect.
- An index build with the CONCURRENTLY option failed, leaving an "invalid" index.

## **Maintain SQLite**

### **Maintain Database and Table**

#### **Analyze Database and Analyze Tables**

Gather statistics about tables and indexes and stores the collected information in internal tables of the database where the query optimizer can access the information and use it to help make better query planning choices.

#### **Vacuum Database**

Rebuild the entire database. VACUUM only works on the main database. It is not possible to VACUUM an attached database file.



### **Reindex Database and Reindex Tables**

Delete and recreate all indexes in database or attached to the tables from scratch. This is useful when the definition of a collation sequence has changed.

### **Maintain Index**

#### **Reindex**




Delete and recreate the index from scratch. This is useful when the definition of a collation sequence has changed.

# Table Viewer

**Table Viewer** displays the table data as a grid. Data can be displayed in two modes: **Grid View** and **Form View**.

The toolbars of Table Viewer provides the following functions for managing data:

- **Begin Transaction/Commit/Rollback**

Click  **Begin Transaction** to start a transaction. To make permanent all changes performed in the transaction, click  **Commit**. Or, click  **Rollback** to undo work done in the current transaction.

**Hint:** The **Commit** and **Rollback** buttons are available only when **Auto Begin Transaction** is enabled under [Options](#) or after clicking the **Begin Transaction** button.

- [Edit TEXT/BLOB/BFile](#)

Allow you to view and edit the content of TEXT, BLOB and BFile fields.

**Note:** Only Oracle supports BFile.

- [Filter Data](#)

Allow you to filter records by creating and applying filter criteria for the data grid.

- [Sort Records](#)

Sort Records by custom order.


- [Import Data](#)

Import data from files.

- [Export Data](#)

Export data to files.

## Grid View

The  Grid View allows you to view, update, insert, or delete data in a table. The pop-up menu of the grid provides the following additional functions: set the field value as Null/Empty String, use current field value as a filter, format grid view, and more.

## Using Navigation Bar

**Table Viewer** provides a convenient way to navigate among the records/pages using **Record/Page Navigation Bar** buttons. All buttons are used to navigate left and right to the previous or the next records/pages.



### Record Navigation Bar

Button	Description
	New record: enter a new record. At any point when you are working with your table in the grid view, click on this button to get a blank display for a record.
	Delete record: delete an existing record.
	Apply changes: apply the changes.
	Cancel changes: remove all edits made to the current record.
	Refresh: refresh the table.
	Stop: stop when loading enormous data from server.

### Extra Record Navigation Bar Button for Form View

Button	Description
	First Record: move to the first record.
	Previous Record: move one record back (if there is one) from the current record.
	Next Record: move one record ahead.
	Last Record: move to the last record.

**Note:** The SQL statement shows under the Record Navigation Bar indicates any statement has just been executed.

### Page Navigation Bar

Button	Description
	First Page: move to first page.
	Previous Page: move to previous page.
	Next Page: move to next page.
	Last Page: move to last page.
	Limit Record Setting: set number of records showing on each page.
	Grid View: switch to grid view.
	Form View: switch to form view.

Use the **Limit Record Setting** button to enter to the edit mode.

**Limit Records** ☐ **records per page**

Check this option if you want to limit the number of records showed on each page. Otherwise, all records will be displayed in one single page. And, set the **records per page** value in the edit field. The number representing the number of records showed per page.

**Note:** This setting mode will take effect on current table only. To adjust the global settings, see [Options](#).

						<input checked="" type="checkbox"/> Limit Record <input type="text" value="1000"/> records per page		
SELECT * FROM 'customer' LIMIT 0, 1000						Record 1 of 599 in page 1		

### Record a of b in page c





The Record/Page Indicator displays the numbers representing the selected record and page.

- a. the selected record.
- b. number of records in the current page.
- c. the current page.


## Editing Records

The navigation bar allows you to switch the records quickly, insert, update or delete records. View data as a grid is most helpful for entering new records and editing old records in a table.

To add a record

1. Make sure that your cursor is situated in the first blank cell on the table, then enter the desired data. If you are adding the new record into an existing table, just simply click on an existing record and click the  from the navigation bar or press CTRL+N to get a blank display for a record.
2. Watch the graphics symbol in the record selectors box just to the left of your record. It will change from the , which indicates that it is the current record, to , which indicates that you are editing this record.
3. Just simply move to another record to save the record or click the  from the navigation bar.

To edit a record

1. Select the record that you wish to edit by clicking in the specific field you want to change.
2. Type in the new data for that field.
3. Just simply move to another record, the new data will overwrite the previous data or click the  from the navigation bar.


**Note:** Close the table is another way to save the records.

To edit multiple cells with same data

1. Select a block of cells in the data grid.
2. Type in the new data.

**Note:** Changes will apply to multiple fields with compatible data type.

To delete a record

1. Select the record that you wish to delete.
2. Just simply right-click and select **Delete Record** or click the  from the navigation bar.

## Edit Records with Special Handling

To set **Empty String** for the cell, right-click the selected cell and select **Set to Empty String**.

To set **Null** value for the cell, right-click the selected cell and select **Set to NULL**.

To edit the text field record, just simply click **Show Memo In Grid** from the **View** menu.




**Note:** Available only for MySQL, Oracle, PostgreSQL and MariaDB.

customer_id	store_id	first_name	last_name
1	1	(WIDEMEMO)	SMITH
2	1	(WIDEMEMO)	JOHNSON
3	1	(WIDEMEMO)	WILLIAMS
4	2	(WIDEMEMO)	JONES
5	1	(WIDEMEMO)	BROWN


**Hint:** To view/edit the text field record in an ease way, see [Memo Editor](#).

To view images in the grid, just simply click **Show Image In Grid** from the **View** menu.

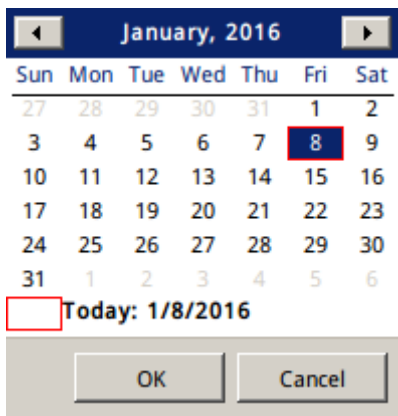
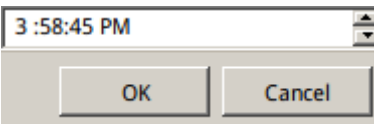
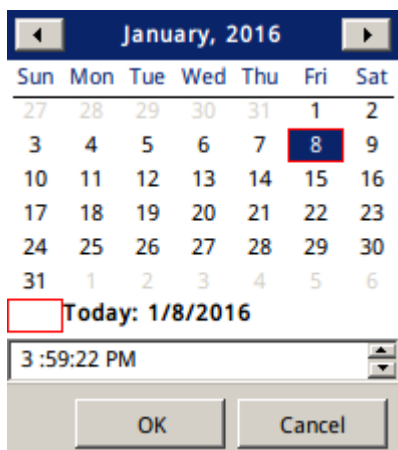
**Note:** Available only for MySQL, Oracle, PostgreSQL and MariaDB.

id	image
1	
2	
3	

**Hint:** To view/edit the image in an ease way, see [Image Editor](#).

To edit a Date/Time record, just simply click  or press CTRL+ENTER to open the editor for editing. Choose/enter the desired data. The editor used in cell is determined by the field type assigned to the column.

**Note:** Available only for MySQL, Oracle, PostgreSQL and MariaDB.


Date	Time	DateTime/Timestamp
		

To edit an Enum record, just simply choose the record from the drop-down list.


**Note:** Available only for MySQL, PostgreSQL and MariaDB.

id	course
1	Maths
2	

Chinese  
English  
Maths  
Music  
Sports

To edit a Set record, just simply click  or press CTRL+ENTER to open the editor for editing. Select the record(s) from the list. To remove the records, uncheck them in the same way.

**Note:** Available only for MySQL and MariaDB.

id	course
1	English, Maths, Music, Sports
2	Apply Maths, Sports 

☐ Chinese  
☐ English  
☐ Maths  
☐ Music  
☒ Apply Maths  
☒ Sports

OK Cancel

To view BFile content, just simply enable **Preview BFile** under the **View** menu.


**Note:** Available only for Oracle.

To generate UUID/GUID, right-click the selected cell and select **Generate UUID**.

**Note:** Available only for PostgreSQL.

### Edit Records with Foreign Key (Foreign Key Data Selection - Available only in Full Version)

**Foreign Key Data Selection** is a useful tool for letting you to get the available value from the reference table in an easy way. It allows you to show additional record(s) from the reference table and search for a particular record(s).


To include data to the record, just simply click  or press CTRL+ENTER to open the editor for editing.

3	3	...	1	1185	5.99	2005-06-15 00:54:12
4	fk_payment_customer : payment (customer_id) - customer (customer_id)					
5	customer_id	store_id	first_name			
6		1	1	MARY		
7		2	1	PATRICIA		
8		3	1	LINDA		
9		4	2	BARBARA		
10		5	1	ELIZABETH		
11		6	2	JENNIFER		
12		7	1	MARIA		
13		8	2	SUSAN		
14		9	2	MARGARET		
15						
16						
17						
18						
19						



Just simply double-click to select the desired data.

**Hint:** By default, the number of records showed per page is **100**. To show all records, right-click anywhere on the grid and select **Show All**. To adjust the global settings, see [Options](#).

To refresh the record, right-click anywhere on the grid and select **Refresh** or press F5.

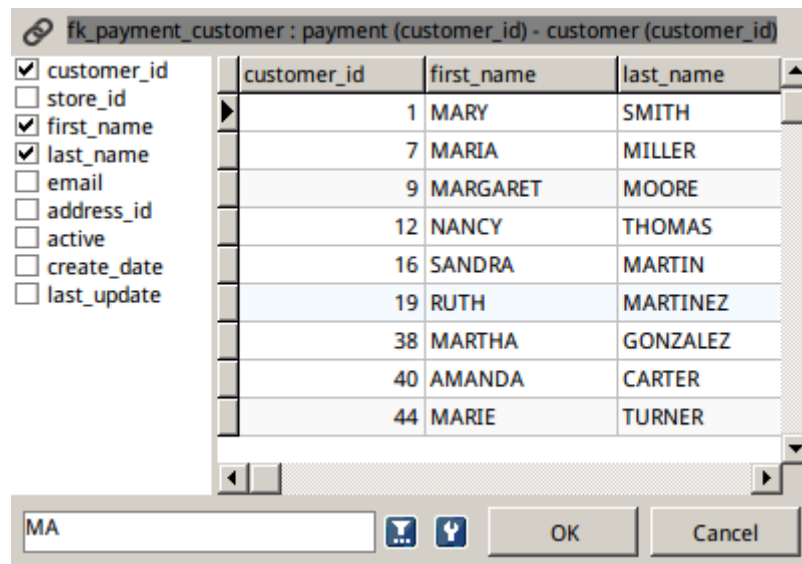
Click  to open a panel on the left for showing a list of column name(s). Just simply click to show the additional column. To remove the column(s), uncheck them in the same way.

fk_payment_customer : payment (customer_id) - customer (customer_id)			
<input checked="" type="checkbox"/> customer_id	customer_id	first_name	last_name
<input type="checkbox"/> store_id			
<input checked="" type="checkbox"/> first_name	1	MARY	SMITH
<input checked="" type="checkbox"/> last_name	2	PATRICIA	JOHNSON
<input type="checkbox"/> email	3	LINDA	WILLIAMS
<input type="checkbox"/> address_id	4	BARBARA	JONES
<input type="checkbox"/> active	5	ELIZABETH	BROWN
<input type="checkbox"/> create_date	6	JENNIFER	DAVIS
<input type="checkbox"/> last_update	7	MARIA	MILLER
	8	SUSAN	WILSON
	9	MARGARET	MOORE

**Hint:** To set column in ascending or descending mode, right-click anywhere on the column and select **Sort** ->  **Sort Ascending** /  **Sort Descending**.

To find for the text in the editor window, right-click anywhere on the grid and select **Find** or press CTRL+F.

Enter a value into the edit box and click  to filter for the particular record(s).



**Hint:** To remove the filter results, right-click anywhere on the grid and select **Show All**.

### Copy Data from Navicat


Data that being copied from Navicat goes into the clipboard with the fields delimited by tabs and the records delimited by carriage returns. It allows you to easily paste the clipboard contents into any application you want. Spreadsheet applications in general will notice the tab character between the fields and will neatly separate the clipboard data into rows and columns.

To select data using **Keyboard Shortcuts**

CTRL+A	Toggle the selection of all rows and columns in a data grid.
SHIFT+ARROW	Toggle the selection of cells as you move up/down/left/right in the data grid.

To select data using **Mouse Actions**

- Select the desired records by holding down the CTRL key while clicking on each row.
- Select a block of cells.

**Note:** After you have selected the desired records, just simply press CTRL+C or right-click and select the  **Copy** from the pop-up menu.


### Paste Data into Navicat

Data is copied into the clipboard will be arranged as below format:

1. Data is arranged into rows and column.
2. Rows and columns are delimited by carriage returns/tab respectively.
3. Columns in the clipboard have the same sequence as the columns in the data grid you have selected.

When pasting data into Navicat, you can replace the contents of current records and append the clipboard data into the table. To replace the contents of current records in a table, you must select the cells in the data grid whose contents



must be replaced by the data in the clipboard. Just simply press CTRL+V or right-click and select the  **Paste** from the pop-up menu. Navicat will paste all the content in the clipboard into the selected cell. The paste action cannot be undone if you do not enable transaction.

### Copy Records as Insert/Update Statements

To copy records as Insert/Update statement, select the records and click **Edit -> Copy As -> Insert Statement** or **Update Statement** from the menu. Then, you can paste the statements in any editor.

### Copy Field Name

To copy field names as tab separated values, select the columns/data and choose **Edit -> Copy As -> Tab Separated Values (Field Name only)** from the menu. If you want to copy data only or both field names and data, you can choose **Tab Separated Values (Data only)** or **Tab Separated Values (Field Name and Data)** respectively.

### Save Data as a File

You can save the data in the table grid to a file. Simply right-click a cell and choose **Save Data As**. Enter the file name and file extension in the Save As dialog.

**Note:** Not available when multiple selection.

## Sorting/Finding/Replacing Records

### Sorting Records


Server stores records in the order they were added to the table. Sorting in Navicat is used to temporarily rearrange records, so that you can view or update them in a different sequence.

Move over the column caption whose contents you want to sort by, click the right side of the column and select the **Sort Ascending**, **Sort Descending** or **Remove Sort**.

	customer_id	store_id	first_name	last_name	email
▶	1	1	MARY	Sort Ascending	MARY.SMITH@sakilacustomeri
	2	1	PATRICIA	Sort Descending	PATRICIA.JOHNSON@sakilacu
	3	1	LINDA	Remove Sort	LINDA.WILLIAMS@sakilacusto
	4	2	BARBARA		BARBARA.JONES@sakilacustor
	5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacus
	6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustor

To sort by custom order of multi fields, click the  **Sort** from the toolbar.

### Finding Records

The **Find** bar is provided for quick searching for the text in the editor window. Just simply click **Edit ->  Find** from the menu or press CTRL+F. Then, choose **Find Data** and enter a search string.

customer @sakila (MySQL 5.7) - Table

File Edit View Window Help

Begin Transaction Memo Filter Sort Import Export

customer_id	store_id	first_name	last_name	email	address_id	active
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer	5	1
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacu	6	1
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustc	7	1
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustor	8	1
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacus	9	1
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustor	10	1
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustom	11	1
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustor	12	1
9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacus	13	1
10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacust	14	1
11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustoi	15	1
12	1	NANCY	THOMAS	NANCY.THOMAS@sakilacusto	16	1
13	2	KAREN	JACKSON	KAREN.JACKSON@sakilacustor	17	1
14	2	BETTY	WHITE	BETTY.WHITE@sakilacustome	18	1
15	1	HELEN	HARRIS	HELEN.HARRIS@sakilacustom	19	1
16	2	SANDRA	MARTIN	SANDRA.MARTIN@sakilacustc	20	0
17	1	DONNA	THOMPSON	DONNA.THOMPSON@sakilacu	21	1
18	2	CAROL	GARCIA	CAROL.GARCIA@sakilacustom	22	1
19	1	RUTH	MARTINEZ	RUTH.MARTINEZ@sakilacusto	23	1
20	2	SHARON	ROBINSON	SHARON.ROBINSON@sakilacu	24	1
21	1	MICHELLE	CLARK	MICHELLE.CLARK@sakilacusto	25	1
22	1	LAURA	RODRIGUEZ	LAURA.RODRIGUEZ@sakilacus	26	1

Find Data: MA Next Highlight All Replace

SELECT \* FROM 'customer' LIMIT 0, 1000 Record 16 of 599 in page 1

The search starts at the cursor's current position to the end of the file. There will not have differentiates when performing a uppercase or lowercase search.

To find for the next text, just simply click **Next** or press F3.

## Replacing Records

To open the **Replace** bar, simply check the **Replace** box and enter the text you want to search and replace.

Click **Replace** or **Replace All** button to replace the first occurrence or all occurrences automatically.

Find Data: MA Next Highlight All ☒ Replace

SA Replace Replace All

If you clicked **Replace All** button, you can click **Apply** button to apply the changes or **Cancel** button to cancel the changes.


Find: ic Next Highlight All ☒ Replace

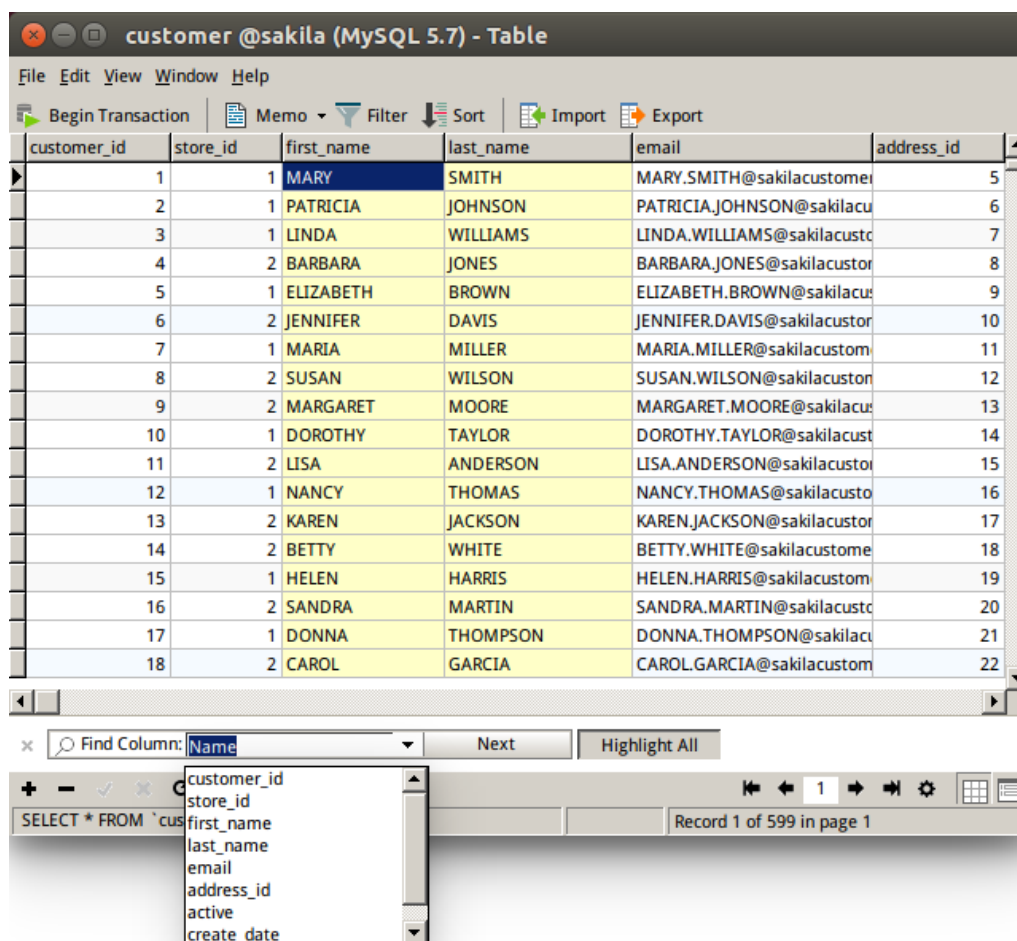
Highlight All Incremental Search Match Case Regular Expression Whole Word

Replace Replace All

Query time: 0.000s

## Finding Columns



To search a column, just simply click **Edit** ->  **Find** from the menu or press CTRL+F. Then, choose **Find Column** and enter a search string.



customer_id	store_id	first_name	last_name	email	address_id
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer	5
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacu	6
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacusto	7
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustor	8
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacus	9
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustor	10
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustom	11
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustom	12
9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacus	13
10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacust	14
11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustor	15
12	1	NANCY	THOMAS	NANCY.THOMAS@sakilacusto	16
13	2	KAREN	JACKSON	KAREN.JACKSON@sakilacustor	17
14	2	BETTY	WHITE	BETTY.WHITE@sakilacustome	18
15	1	HELEN	HARRIS	HELEN.HARRIS@sakilacustom	19
16	2	SANDRA	MARTIN	SANDRA.MARTIN@sakilacustc	20
17	1	DONNA	THOMPSON	DONNA.THOMPSON@sakilacu	21
18	2	CAROL	GARCIA	CAROL.GARCIA@sakilacustom	22

## Filtering Records

Use either of the following methods to filter the data in the grid:

- Right-click a field and select the **Filter** from the pop-up menu to filter records by the current value of the selected column.
- The **Custom Filter** Dialog is provided for quick building a simple filter. Just simply right-click a field and select the **Filter** -> **Custom Filter** from the pop-up menu. Use character '\_' to represent any single symbol in the condition and use character '%' to represent any series of symbols in the condition.
- You can also customize your filter in a more complicated way by right-click a field and selecting the **Filter** ->  **Filter** from the pop-up menu or clicking the  **Filter** from the toolbar. The Filter Wizard becomes visible at the top of grid, where you can see the active filtering condition and easily enable or disable it by clicking a check box at the left.

## Manipulating Raw Data

Navicat normally recognize what user has input in grid as normal string, any special characters or functions would be processed as plain text (that is, its functionality would be skipped).

Editing data in **Raw Mode** provides an ease and direct method to apply server built-in function. To access the Raw Mode function, just simply select **View -> Raw Mode** from the menu.

**Note:** Available only for MySQL, PostgreSQL, SQLite and MariaDB.

	customer_id	store_id	first_name	last_name
	'1'	'1'	'MARY'	'SMITH'
	'2'	'1'	'PATRICIA'	'JOHNSON'
	'3'	'1'	'LINDA'	'WILLIAMS'
I	'4'	'2'	CONCAT('BARBARA', ' ', 'A')	'JONES'
	'5'	'1'	'ELIZABETH'	'BROWN'

## Formatting Table Grid

Use the following methods to format the table grid:

### Move Columns

1. Click on the column header and hold down the left mouse button.
2. Move the pointer until a double black line appears in the desired location.
3. Release the mouse and the column will move.

	customer_id	store_id	first_name	last_name
	1	1	MARY	SMITH
	2	1	PATRICIA	JOHNSON
	3	1	LINDA	WILLIAMS
▶	4	2	BARBARA	JONES
	5	1	ELIZABETH	BROWN
	6	2	JENNIFER	DAVIS
	7	1	MARIA	MILLER

### Freeze Selected Column

If there are many columns in the table and you want to freeze one or more columns to identify the record. Just simply right-click the column you want to freeze and select **Display -> Freeze Selected Column** or select **View -> Freeze Selected Column** from the menu.

The frozen column(s) will move to the leftmost position in the table grid. This action will locks the frozen column(s), preventing them from being edited.

To unfreeze the columns, just simply right-click anywhere on the table grid and select **Display -> Unfreeze Columns** or select **View -> Unfreeze Columns** from the menu.

## Set Display Format

The **Set Display Format** Dialog is provided for you to customize format applied to exported data on the selected column. Just simply right-click the column you want to edit its format and select **Display -> Set Display Format** or select **View -> Set Display Format** from the menu. Edit the format style to adjust the result format in the way you need. For example: dd-mm-yyyy.

**Hint:** This action applies on the selected column only. To adjust the global settings, see [Options](#).

## Set Column Width

Click right border at top of column and drag either left or right.

Double-click right border at top of column to obtain the best fit for the column.

Right-click the column you want to set the column width with and select **Display -> Set Column Width** or select **View -> Set Column Width** from the menu. Specify width in the **Set Column Width** Dialog. The default value is 120.

**Hint:** The result only applies on the selected column. To adjust the global settings, see [Options](#).

## Set Row Height

Right-click anywhere on the table grid and select **Display -> Set Row Height** or select **View -> Set Row Height** from the menu. Specify row height in the **Set Row Height** Dialog. The default value is 17.

**Hint:** This action applies on the current table grid only. To adjust the global settings, see [Options](#).

## Show/Hide Columns

If there are many columns in the table and you want to hide some of them from the table grid. Just simply right-click anywhere on the table grid and select **Display -> Show/Hide Columns** or select **View -> Show/Hide Columns** from the menu. Select the columns that you would like to hide.

The hidden column(s) will disappear from the table grid.

To unhide the columns, just simply right-click anywhere on the table grid and select **Display -> Show/Hide Columns** or select **View -> Show/Hide Columns** from the menu. Select the columns that you would like to redisplay.

<input checked="" type="checkbox"/> customer_id	customer_id	store_id	last_name	address_id
<input checked="" type="checkbox"/> store_id	1	1	SMITH	5
<input type="checkbox"/> first_name	2	1	JOHNSON	6
<input checked="" type="checkbox"/> last_name	3	1	WILLIAMS	7
<input type="checkbox"/> email	4	2	JONES	8
<input checked="" type="checkbox"/> address_id	5	1	BROWN	9
<input checked="" type="checkbox"/> active	6	2	DAVIS	10
<input checked="" type="checkbox"/> create_date				
<input checked="" type="checkbox"/> last_update				


## Show/Hide ROWID

If you want to display or hide the rowid (address) of every row, right-click anywhere on the table grid and select **Display** -> **Show/Hide ROWID** or select **View** -> **Show/Hide ROWID** from the menu.

The column **ROWID** will be showed in the last column.

**Note:** Available only for Oracle and SQLite.

## Form View (Available only in Full Version)

The  Form View allows you to view, update, insert, or delete data as a form, which the current record is displayed: field name and its value. The pop-up menu of the form provides the following additional functions: set the field value as Null/Empty String, use current field value as a filter, format form view, and more.

The navigation bar allows you to switch the records quickly, insert, update or delete records.

Related topic:





[Sorting/Finding/Replacing Records](#)

[Filtering Records](#)


[Manipulating Raw Data](#)


[Formatting Table Grid](#)

## Assistant Editor




Navicat provides Memo/Hex/Image/Dynamic Column pane to view and edit TEXT/BLOB/BFile fields content. The editor allows you to view, update, insert, or delete data in a table. Click  **Memo**,  **Hex**,  **Image** and  **Dynamic Column** from the toolbar to activate the appropriate viewer/editor.



**Note:** Oracle BFile fields cannot be edited.

The **Memo** pane allows you to edit data as a simple text. Use the  button on the navigation bar to update the changed records to the table.


The **Hex** pane allows you to edit data in hexadecimal mode. Use the  button on the navigation bar to update the changed records to the table.

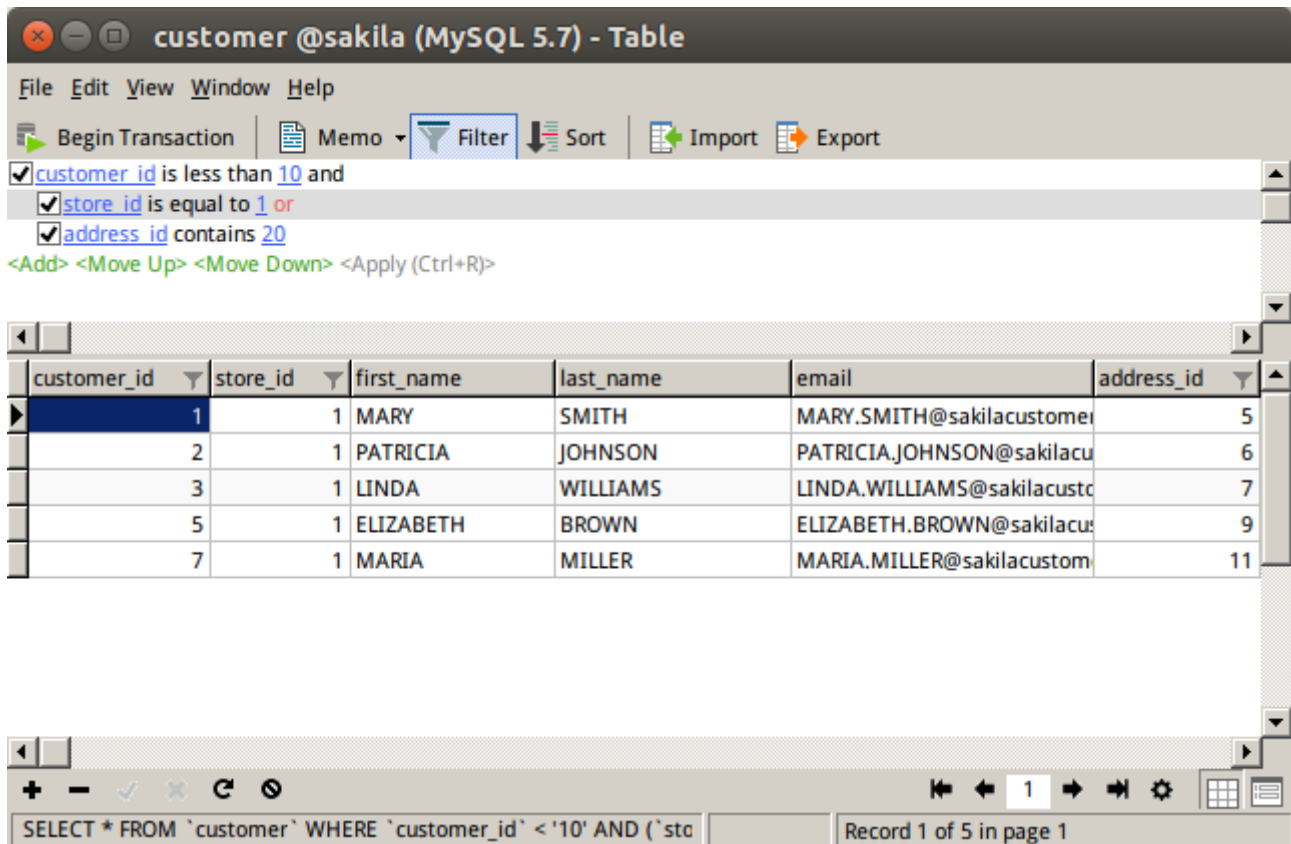
**Note:** Use the INSERT key on the keyboard to switch between Insert and Overwrite modes.

The **Image** pane allows you to show data as image. Use the  **Load**,  **Save to disk** and  **Clear** button to load/remove the image from a file, or save the image to a file.

The **Dynamic Column** pane allows you to edit data as dynamic column in MariaDB. Use the  and  buttons on the left to add and delete values.

## Filter Wizard

Filter Wizard allows you to facilitate creating and applying filter criteria that you specify for the table grid. Moreover, it allows you to save filter criteria as a profile for future use. Click  **Filter** from the toolbar to activate the editor.



1. To add a new condition to the criteria, just simply click **<Add>**.
2. Click on the column box (next to the checkbox) and choose a table column. You can choose **[Additional Filter]** from the list to enter the condition manually.
3. Click on the operator box (next to the column box) and choose a filter operator.

Filter Operator	Result
is equal to <?>	My_Field = 'your_value'
is not equal to <?>	My_Field <> 'your_value'
is less than <?>	My_Field < 'your_value'
is less than or equal to <?>	My_Field <= 'your_value'
is greater than <?>	My_Field > 'your_value'
is greater than or equal to <?>	My_Field >= 'your_value'
contains <?>	My_Field LIKE '%your_value%'
contains (case insensitive) <?> <b>Note:</b> Available only for PostgreSQL.	My_Field ILIKE '%your_value%'
does not contain <?>	My_Field NOT LIKE '%your_value%'
does not contain (case insensitive) <?> <b>Note:</b> Available only for PostgreSQL.	My_Field NOT ILIKE '%your_value%'

begin with <?>	My_Field LIKE 'your_value%'
end with <?>	My_Field LIKE '%your_value'
is null	My_Field IS NULL
is not null	My_Field IS NOT NULL
is empty	My_Field = ''
is not empty	My_Field <> ''
is between <?> <?>	((My_Field >= your_value1) AND (My_Field <= your_value2))
is not between <?> <?>	NOT ((My_Field >= your_value1) AND (My_Field <= your_value2))
is in list <?>	My_Field IN ('aaa','bbb',...)
is not in list <?>	My_Field NOT IN ('aaa','bbb',...)



- Click on the criteria values box (next to the operator box) to activate the appropriate editor and enter the criteria values. The editor used in criteria values box is determined by the editor type assigned to the corresponding column.
- Click on the logical operator box (next to the criteria values box) and choose **and** or **or**.
- Repeat step 1-5 to add another new condition.
- Click **<Apply (Ctrl+R)>** or press CTRL+R to see the result of the filtering you made.

**Hint:** To set compound filter, simply right-click on a selected condition and choose **Indent** or **Outdent**.






You are allowed to save filter criteria to and load them from the profiles for future use. Just simply right-click on the Filter Wizard and select **Open Profile**, **Save Profile**, **Save As** or **Delete Profile**.



# Query

A query is used to extract data from the database in a readable format according to the user's request. Navicat provides two powerful tools for working with the SQL queries: Query Editor for editing the query text directly and Query Builder for building queries visually. You can save your queries for setting [schedule](#). Click  to open an object list for **Query**. Or, you can simply click  button in the main window to open the Query Builder or the Query Editor.

**Hint:** Queries(.sql) are saved under the [Settings Location](#). If the connection is synchronized to [Navicat Cloud](#), queries are stored in the Cloud.

Button	Description
 Run	Execute the query: Run, Run Selected or Run a statement from here.
 Stop	Stop the query.
 Explain	Show the Query Plan of the query.
 Beautify SQL	Format the codes with the Beautify SQL settings in Editor.
 <a href="#">Export Result</a>	Export the result of the query.

## Query Builder (Available only in Full Version)

Navicat provides a useful tool called **Query Builder** for building queries visually. It allows you to create and edit queries without knowledge of SQL. The database objects are displayed in left pane. Whereas in the right pane, it is divided into two portions: the upper **Diagram Design** pane, and the lower **Syntax** pane.

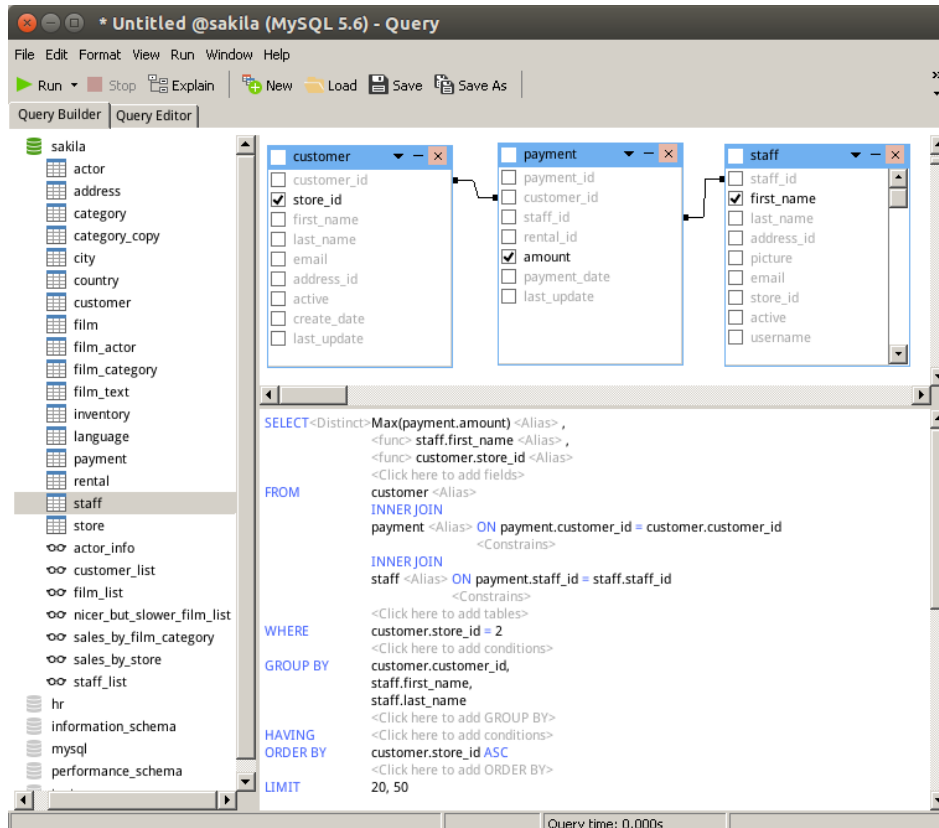
**Note:** Query Builder supports *SELECT* statement only. Use Query Editor for creating complex queries.

Drag a table or a view from the left pane to the Diagram Design pane or double-click it to add it to query. To include a field in the query, check the left of the field name in the Diagram Design pane. To include all the fields, click at the left of the object caption.

To remove the object from the Diagram Design pane, click the cross button at the object caption.

To add the table/view alias, simply double-click the table/view name and enter the alias in the Diagram Design pane.

**Hint:** You are also allowed to set criteria by right-click any fields from the Diagram Design pane.



## Setting Field Association

To associate database objects by two fields, just drag one field from the Object List pane to another and a line will appear between the linked fields.

**Hint:** To delete all the links of some object, click button '-' next to the object alias.

Go to the Syntax pane to change the association between the links, click the operator and choose the properties item from the pop-up menu. You can change the association condition by choosing it from the list (=, <>, <, <=, >, >=). Click **OK** to confirm the changes you made. Also you can change the type of Join.

## Setting Output Fields

The fields you have selected in the Diagram Design pane will be displayed in the Syntax pane which allows you to set their displaying order and modify the output fields of the query using **<Distinct>**, **<func>** and **<Alias>**.

### <Distinct>

Enable this option if you wish the repeated records are not included into the query result.

### <func>

Set the aggregate functions (SUM, MAN, MIX, AVG, COUNT) for each field.

### <Alias>

Change the output query field name.

## Setting Criteria

To add a condition, click the **<--> = <-->** from the **WHERE** clause in the Syntax pane. Click **<-->** to choose the field from the list of all the table fields, available in the query. To define your own criteria, type your values directly in the Edit Tab. Clicking **=** to set condition operator.

### Setting Grouping Criteria

You can set the conditions for grouping query records from the **GROUP BY** clause in the Syntax pane. They are set in the same way as setting criteria. The conditions will be included into the **HAVING** statement of the current query.

### Setting Sorting Criteria

When you query your database/schema, you can sort the results by any field in an ascending or descending order by just adding *ORDER BY* at the end of your query.

In Query Builder, you can set the way of sorting query records from the **ORDER BY** clause in the Syntax pane. To change the sorting direction, click on either **ASC** or **DESC**.

### Setting Limit Criteria

**LIMIT** clause is used to limit your query results to those that fall within a specified range. You can use it to show the first X number of results, or to show a range from X - Y results. It is phrased as Limit X, Y and included at the end of your query. X is the starting point (remember the first record is 0) and Y is the duration (how many records to display).

**Note:** Available only for MySQL, PostgreSQL, SQLite and MariaDB

## Query Editor

Navicat provides a useful tool called **Query Editor** for creating and executing queries. It allows you to create and edit SQL text for a query, prepare and execute selected queries.

**Hint:** Query text will be automatically generated while you build in Query Builder.

You are allowed to run selected portion of query, just simply right-click the highlighted query and select **Run Selected**.

You can define multiple SQL statements in one Editor window, and the editor let you run the current statement your cursor is on (place your cursor in the front of the desired statement). Just simply select **Run a statement from here** or press F7.

**Note:** Select **Run a statement from here** or press F7, the next statement will continue to run.

### Editor Advanced Features

Navicat provides a wide range advanced features, such as compelling code editing capabilities, smart code-completion, sql formatting, and more.

## SQL Formatting

To change the SQL statement format, simply choose from the **Format** menu -

### Indent

Increase/decrease indent for the selected lines of codes.

### Comment

Comment/uncomment the selected lines of codes.

### Convert Case

Format the selected codes into upper/lower case.

### Beautify SQL (Available only in Full Version)

Format the selected codes with the Beautify SQL settings.

### Beautify SQL With (Available only in Full Version)

Change the Beautify SQL settings.

Option	Description
Use tab character	Check this option to use tab character.
Tab size	Set the tab size.
Short brace length	Set the length of the short brace.
Upper case keywords	Format all the SQL keywords to upper case.
Save settings	Save the SQL beautify options settings after you click <b>Beautify</b> button.

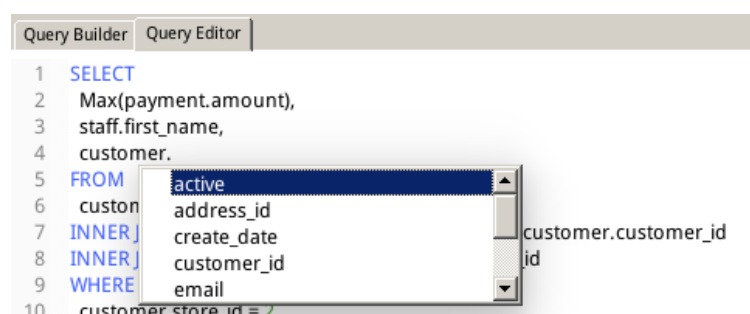
### Minify SQL (Available only in Full Version)

Minify the format of the SQL in the SQL Editor.

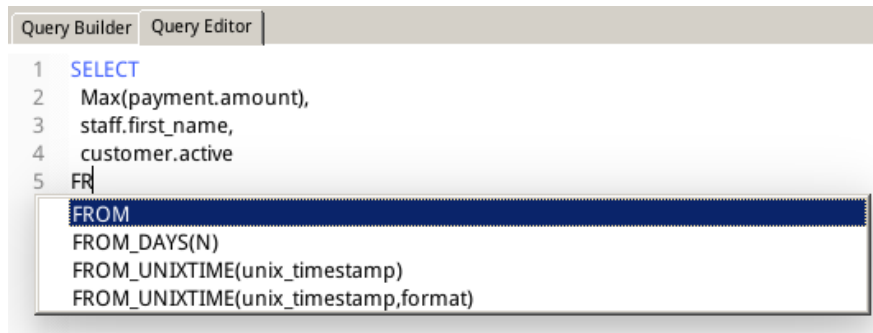
## Code-Completion (Available only in Full Version)

Code-completion in Navicat displays information in drop-down lists as you type your SQL statement in the editor, it assists you with statement completion and the available properties of database objects, for example databases, tables, fields, views etc with their appropriate icons.

To activate the code-completion, just simply press '.' for the available properties of database object currently in the scope.



**Hint:** You may invoke code-completion by typing two characters or pressing CTRL+SPACE on your keyboard for SQL keywords.



**Hint:** Smart code-completion will pop up a list of variants for the word completion automatically.





**Note:** Code-completion can be also applied on View, Functions/Procedures, etc.

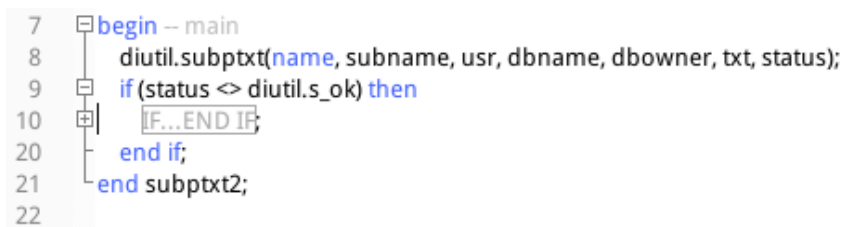
Related topic:

[Code Insight Options](#)

## Code Folding

Code folding feature enables you to collapse blocks of code such that only the first line of the block appears in **Editor**.

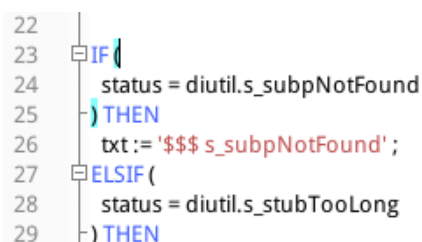
A block of code that can be folded is indicated by an icon  to the left of the first line of the block. A vertical line extends from the icon to the bottom of the foldable code. In contrast, a folded block of code is indicated by an icon  to left of the code block. You can fold the block by clicking  or expand it by clicking  in **Editor**.



## Brace Highlight

Navicat supports to highlight the matching brace in the editor, i.e. ().

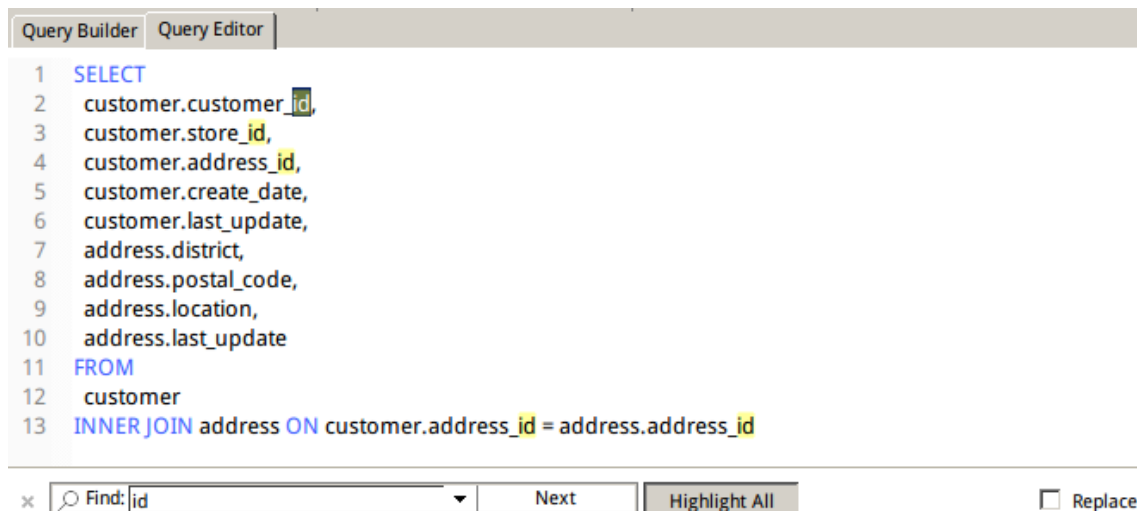
**Note:** The cursor must be on a brace to show the highlight.



## Find and Replace

## Find

The **Find** Dialog is provided for quick searching for the text in the editor window. Just simply click **Edit -> Find** from the menu or press CTRL+F and enter a search string.



The search starts at the cursor's current position to the end of the file. There will not have differentiates when performing a uppercase or lowercase search.

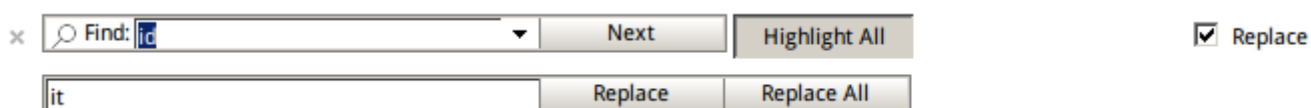
To find for the next text, just simply click **Next** or press F3.

## Replace

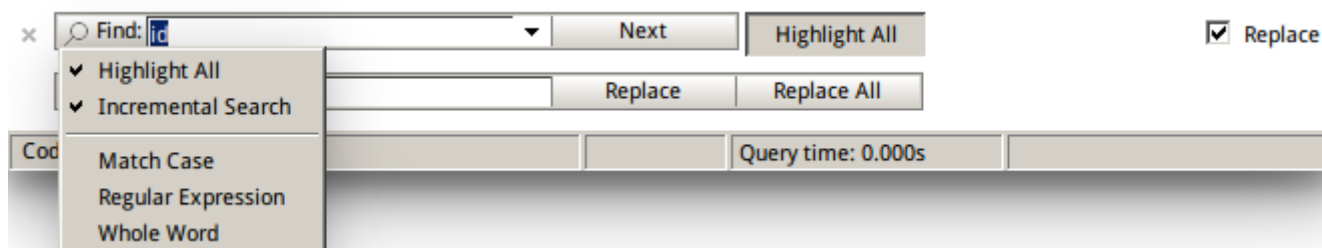
To open the **Replace** Dialog, simply check the **Replace** box and enter the text you want to search and replace.

Click **Replace** button to replace the first occurrence.

Click **Replace All** button to replace all occurrences automatically.



There are some additional **Options** for Find and Replace:



## Copy with Quotes

To copy the SQL statement with quotes, just simply right-click the highlighted SQL. Then, select **Copy with quotes** and choose the format.

**Note:** Only available in Query, View and Materialized View.

### Zoom In/Zoom Out


Navicat has the ability to zoom in or zoom out the SQL in the editor. The zooming options are available from the **View** menu. The same effect can be achieved with keyboard shortcuts.

- Zoom In: [CTRL+=]
- Zoom Out: [CTRL+-]
- Reset: [CTRL+0]

**Hint:** Range from -10 to +20.

**Note:** Files are opened in different tabs will not be effected by the zoom.

## Query Results

To run the query, click  **Run** on the toolbar. If the query statement is correct, the query executes and, if the query statement is supposed to return data, the **Result** tab opens with the data returned by the query. If an error occurs while executing the query, execution stops, the appropriate error message is displayed.

The **Result** tab displays the result data, returned by the query, as a grid. Data can be displayed in two modes: **Grid View** and **Form View**. See [Table Viewer](#) for details.

**Hint:** Navicat supports to return 10 resultsets.

**Note:** You can choose to show the Result tab below the editor or in a new tab by selecting **View -> Show Result -> Below Query Editor** or **In a New Tab**.


### Naming Result Tab

To name the result tab, simply add `-- NAME:tab_name` or `/*NAME:tab_name*/` before each SELECT statement in the Query Editor.

Example:

```
-- NAME:Q1
SELECT * from table1;
/*NAME:Q2*/
SELECT * from table2;
```

### Query Profile and Status (Available only for MySQL and MariaDB)

To show the profile and status when running the query, simply choose **View -> Show Profile and Status** and click  **Run** on the toolbar.

The **Profile** tab displays the query profile: Table lock, System lock, Statistic, etc.

**Note:** For MySQL 5.0, support from 5.0.37 or above.

For MySQL 5.1, support from 5.1.24 or above.

The **Status** tab displays the query status: Bytes received, Bytes sent, etc.

## Query Parameters

Query Builder and Query Editor both support using of parameters inside the query text. You can set query parameters to add variable values to a query each time you run it. The parameter should appear as an identifier with \$ at its beginning, quote with [ ], e.g. [\$any\_name].

Execute the query and the **Input Parameter** Dialog is provided for you to enter the desired data you wish to search.


## Debugging Oracle Query (Available only in Full Version)

To debug the Oracle query click  **Debug** on the toolbar to launch the [Oracle Debugger](#).

Enter the parameter(s) if the query has input parameter(s).



# Model (Available only in Navicat Premium and Enterprise Version)

**Model** is a powerful tool for creating and manipulating physical database models. Click  to open an object list for **Model**. Some of key features are listed here:

- Create and manipulate a physical model graphically.
- Reverse engineer a database/schema, table(s) or view(s) to a physical model.
- Forward engineer a physical model to a sql file or database/schema.
- Create and edit table structures directly.

To create a model, click  **New Model** from the object list toolbar. The New Model window will pop up for you to select the target **Database** and **Version**.

**Hint:** Model files(.ndm) are saved under the [Profiles Location](#). If the model is synchronized to [Navicat Cloud](#), it is stored in the Cloud.

## Model Sidebar

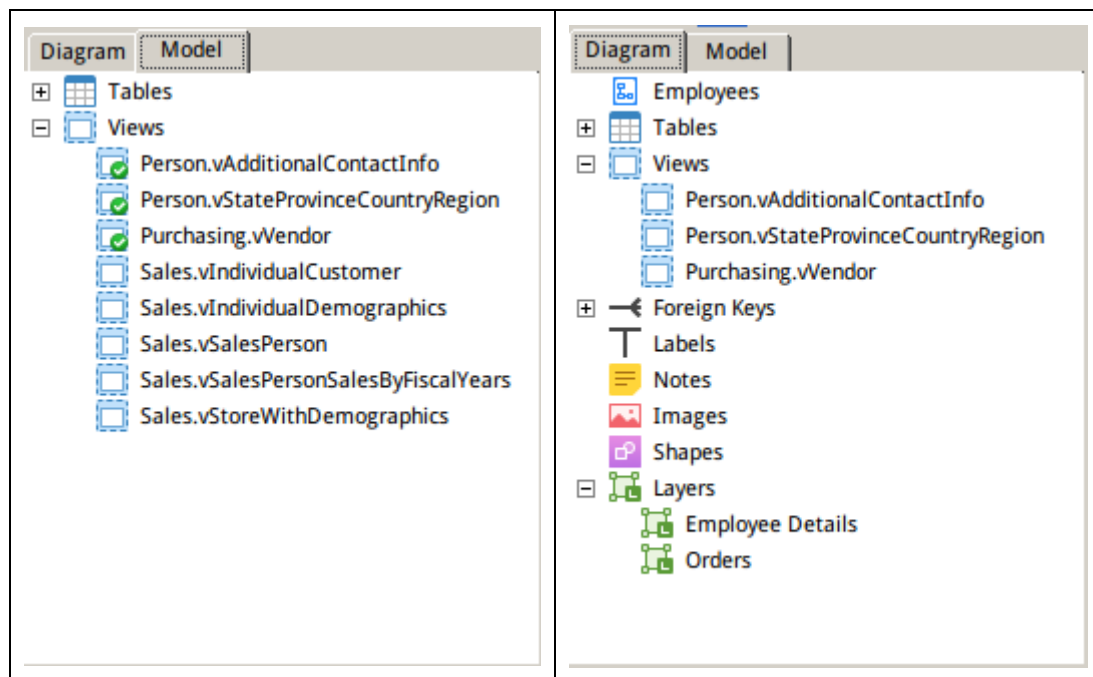
In the model's sidebars, all objects of your model/diagram(s), their properties and action history are listed.

The sidebars consist of the following components:

- Explorer
- History
- Properties
- Overview

## Model Explorer Pane

The **Explorer** pane has two tabs: **Model** and **Diagram**. Model tab holds all tables or views in the model, including those used in each individual diagram. You can simply drag an object from the Model tab and drop to the active diagram canvas. Diagram tab holds all the objects (tables, views, foreign keys, layers, notes, images, etc) added to the active diagram. If the Explorer pane is hidden, choose **View** -> **Show Explorer** from the model's menu.



## Model History Pane

The **History** pane shows all the actions that you have taken. Simply click an action to restore that state. If the History pane is hidden, choose **View -> Show Explorer** and **Show History** from the model's menu.

## Model Properties Pane

The **Properties** pane includes the **Model**, **Diagram** and **Object** tabs for setting default properties for your model. You can edit the properties settings of the model, the active diagram and the selected objects quickly. If the Properties pane is hidden, choose **View -> Show Properties** from the model's menu.

### Begin Arrow Style

The style of the arrow's back.

### Black and White

Check this box to change the diagram color to black and white.

### Bold

Check this box or press CTRL+B to bold the table, view, foreign key or shape.

### Border Color

The color of the shape's border.

### Cap Style

The cap style of the line/arrow.

### Case Sensitivity

The case sensitivity of the table or view names. Available only for MySQL and MariaDB physical models.

**Color**

The color of the object.

**Dash Style**

The dash style of the line/arrow.

**Database Type**

The database type of the model.

**Database Version**

The database version of the model.

**Diagram Notation**

The notation of the diagram. The value for this can be Default, Simple, IDEF1X, UML, IE (Crow's Foot) or Classic.

**End Arrow Style**

The style of the arrow's front.

**Font Bold**

Check this box to bold the note/label font.

**Font Color**

The font color of the note, label or layer.

**Font Italic**

Check this box to apply an italic style to the note/label font.

**Font Name**

The font name of the note, label or layer.

**Font Size**

The font size of the note, label or layer.

**Height**

The height of the object.

**Height (pages)**

The height of the diagram (number of papers).

**Join Style**

The join style of the line/arrow.

**Left**

The number of pixels from the object to the left side of the canvas.

**Model Type**

The type of the model.

**Name**

The name of the object.

**Opacity**

The transparency of the image/shape. The value for this can be between 0 to 100. Use 100 for opacity and 0 for transparent.

**Referenced Cardinality**

The foreign key cardinality of referenced (parent) table.

**Referencing Cardinality**

The foreign key cardinality of referencing (child) table.

**Schema Name**

The schema names of the table/view.

**Show Name**

Check this box to show the name of the foreign key or shape.

**Show Schema Name**

Check this box to show the schema names of the tables/views in the diagram.

**Show View Relationships**

Check this box to show the relationship line of the view.

**Table Font Name**

The font name of the tables.

**Table Font Size**

The font size of the tables.

**Top**

The number of pixels from the object to the top of the canvas.

**Visible**

Check this box to show the foreign key lines.

**Width**

The width of the object.

**Width (pages)**

The width of the diagram (number of papers).


## Model Overview Pane

The **Overview** pane displays the whole active diagram in the canvas. To zoom in or zoom out the selected area of the diagram, adjust the slider. If the Overview pane is hidden, choose **View -> Show Properties** and **Show Overview** from the model's menu. Same effect can be achieved with keyboard shortcuts:


Zoom In: [CTRL++] or [CTRL+Mousewheel Up]



Zoom out: [CTRL+-] or [CTRL+Mousewheel Down]

## Diagram Canvas

**Diagram Canvas** consists of a canvas and a toolbar for you to design the diagram, such as adding objects, formatting diagrams and printing models, etc. A model file can have more than one diagram. Each diagram is represented by a tab in the model. To create a new diagram, choose **File ->  New Diagram** from the model's menu.

## Create Tables

To create a new table, click the  button from the diagram toolbar and click anywhere on the canvas. To add an existing table from the Explorer's Model tab, simply drag and drop the selected table from the Model tab to the canvas.

For Default diagram notation, the  icon means the field is a primary key. The  icon indicates that the field serves as an index.

The pop-up menu options of the table object in canvas include:

### Design Table

Edit the table structure in a designer, e.g. fields, indexes, foreign keys, etc. The tabs and options in the designer depend on the diagram database type you are chosen. For the settings of different tabs, see [Server Objects](#).

### Add Related Objects

Add all related objects to the selected table.

### Add Field

Add fields to the existing table.

### Cut

Remove the table from the diagram and put it on the clipboard.

### Copy

Copy the table from the diagram to the clipboard.

### Paste

Paste the content from the clipboard into the diagram.

**Select All Tables**

Select all the tables in the diagram.

**Delete**

Delete a table from the diagram or from both diagram and model.

**Rename**

Change the name of the table.

**Color**

Change the color of the table.

**Size to Fit**

Resize the table automatically to fit its contents.


**Bring to Front**

Bring the table to the foreground.

**Send to Back**

Move the table to the background.

## Create Views

To create a new view, click the  button from the diagram toolbar and click anywhere on the canvas. To add an existing view from the Explorer's Model tab, simply drag and drop the selected view from the Model tab to the canvas.

**Note:** If you right-click the view connector, you can choose to add or delete vertices and change its color, or go to the source view and the target table.

The pop-up menu options of the view object in canvas include:

**Design View**

Edit the view structure in a designer. The tabs and options in the designer depend on the diagram database type you are chosen. For the settings of different tabs, see [Server Objects](#).

**Add Related Objects**

Add all related tables/views to the selected view.

**Cut**

Remove the view from the diagram and put it on the clipboard.

**Copy**

Copy the view from the diagram to the clipboard.

### **Paste**

Paste the content from the clipboard into the diagram.

### **Select All Views**

Select all the views in the diagram.

### **Delete**

Delete a view from the diagram or from both diagram and model.

### **Rename**

Change the name of the view.

### **Color**

Change the color of the view.

### **Size to Fit**

Resize the view automatically to fit its contents.


### **Bring to Front**

Bring the view to the foreground.

### **Send to Back**

Move the view to the background.

## Create Foreign Keys

To add a foreign key, click the  button from the diagram toolbar and drag and drop a field from the child table to the parent table. To show/hide the linked name label, simply check/uncheck the **Show Name** option in Properties pane.

The pop-up menu options of the foreign key object in canvas include:

### **Design Relation**

Edit the foreign key in a designer. The options in the designer depend on the diagram database type you are chosen.

For the settings, see [Server Objects](#).

### **Cardinality on table\_name1**

Set the cardinality on table\_name1: None, One and Only One, Many, One or Many, Zero or One, Zero or Many.

### **Cardinality on table\_name2**

Set the cardinality on table\_name2: None, One and Only One, Many, One or Many, Zero or One, Zero or Many.

### **Add Vertex**

Add a vertex on a foreign key connector.

**Delete Vertex**

Delete a vertex on a foreign key connector.

**Delete All Vertices**

Delete all vertices on a foreign key connector.

**Go to Source**

Go to and select the source (child) table.

**Go to Target**

Go to and select the target (parent) table.

**Paste**

Paste the content from the clipboard into the diagram.

**Select All Relations**

Select all the foreign keys in the diagram.


**Delete from Diagram and Model**

Delete a foreign key from the diagram and model.

**Color**

Change the color of the foreign key.

## Create Labels

Labels are typically used to help document the diagram design process. For example, to explain a grouping table objects. To create a new label, click the  button from the diagram toolbar and click anywhere on the canvas.

The pop-up menu options of the label object in canvas include:

**Edit**

Change the content of the label.

**Cut**

Remove the label from the diagram and put it on the clipboard.

**Copy**

Copy the label from the diagram to the clipboard.

**Paste**

Paste the content from the clipboard into the diagram.

**Select All**

Select all the labels in the diagram.



**Delete**

Delete a label from the diagram.

**Size to Fit**

Resize the label automatically to fit its contents.


**Bring to Front**

Bring the label to the foreground.

**Send to Back**

Move the label to the background.

## Create Notes

Notes are typically used to help document the diagram design process. For example, to explain a grouping table objects. To create a new note, click the  button from the diagram toolbar and click anywhere on the canvas.

The pop-up menu options of the note object in canvas include:

**Edit**

Change the content of the note.

**Cut**

Remove the note from the diagram and put it on the clipboard.

**Copy**

Copy the note from the diagram to the clipboard.

**Paste**

Paste the content from the clipboard into the diagram.

**Select All Notes**

Select all the notes in the diagram.

**Delete**

Delete a note from the diagram.

**Color**

Change the color of the note.

**Size to Fit**

Resize the note automatically to fit its contents.


**Bring to Front**

Bring the note to the foreground.

### **Send to Back**

Move the note to the background.

## Create Images

To create a new image, click the  button from the diagram toolbar and click anywhere on the canvas. Then, select an image file in the Open dialog box

The pop-up menu options of the image object in canvas include:

### **Reset Size**

Reset the size of the image to its original size.

### **Reset Aspect Ratio**

Maintain image original width to height ratio.

### **Cut**

Remove the image from the diagram and put it on the clipboard.

### **Copy**

Copy the image from the diagram to the clipboard.

### **Paste**

Paste the content from the clipboard into the diagram.

### **Select All Images**

Select all the images in the diagram.

### **Delete**

Delete an image from the diagram.


### **Bring to Front**

Bring the image to the foreground.

### **Send to Back**

Move the image to the background.

## Create Shapes

To create a new shape (line, arrow, rectangle, ellipse, user, database, cloud, trigger, server, desktop or mobile), click the  button from the diagram toolbar and choose the type of shape. Then, click anywhere on the canvas. To show/hide the linked name label, simply check/uncheck the **Show Name** option in Properties pane.

The pop-up menu options of the shape in canvas include:

**Reset Aspect Ratio** *(only for rectangle, ellipse, user, database, cloud, trigger, server, desktop and mobile)*

Maintain shape original width to height ratio.

### **Cut**

Remove the shape from the diagram and put it on the clipboard.

### **Copy**

Copy the shape from the diagram to the clipboard.

### **Paste**

Paste the content from the clipboard into the diagram.

### **Select All Shapes**

Select all the shapes in the diagram.

### **Delete**

Delete a shape from the diagram.

### **Color**

Change the color of the shape.

**Border Color** *(only for rectangle, ellipse, user, database, cloud, trigger, server, desktop and mobile)*

Change the color of the shape's border.

**Begin Arrow Style** *(only for arrow)*

Change the style of the arrow's back.

**End Arrow Style** *(only for arrow)*

Change the style of the arrow's front.

**Add Vertex** *(only for line and arrow)*

Add a vertex on a line or arrow.

**Delete Vertex** *(only for line and arrow)*

Delete a vertex on a line or arrow.

**Delete All Vertices** *(only for line and arrow)*

Delete all vertices on a line or arrow.


### **Bring to Front**

Bring the shape to the foreground.

### **Send to Back**

Move the shape to the background.

## Create Layers

Layers are used to help organize objects (e.g. tables, notes, images, etc) on the canvas. You can add all related objects to the same layer. For example, you may choose to add all your sales related tables to one layer. To create a new layer, click the  button from the diagram toolbar and click anywhere on the canvas.

The pop-up menu options of the layer object in canvas include:

### Cut

Remove the layer from the diagram and put it on the clipboard.

### Copy

Copy the layer from the diagram to the clipboard.

### Paste

Paste the content from the clipboard into the diagram.

### Select All Layers

Select all the layers in the diagram.

### Delete

Delete a layer from the diagram.

### Color

Change the color of the layer.

### Size to Fit

Resize the layer automatically to fit its contents.

### Bring to Front

Bring the layer to the foreground.

### Send to Back

Move the layer to the background.

## Format Diagram

### Show Grid

To turn the grid on in the diagram canvas, choose **View -> Show Grid** from the menu.

### Snap to Grid

To align objects on the canvas with the grid, choose **View -> Snap To Grid** from the menu.

### Change Diagram Notation

To change the notation of the diagram, choose **Diagram -> Diagram Notation** and select the notation from the menu.

Default	The default notation style used in Navicat.
Simple	A simple notation style. The table and view objects will only show the name.
IE (Crow's Foot)	Crow's Foot notation style.
IDEF1X	The ICAM DEFinition language information modeling method.
UML	Universal Modeling Language style.
Classic	A classic notation style.
Black and White	Change the color of the diagram to black and white.
Show Schema Name	Show the schema names of the tables and views in the diagram.

### Change Diagram Dimensions

To change the number of pages used in the diagram, choose **Model -> Diagram Dimensions** from the menu and set the Width and Height.

### Align Objects

To align objects on the canvas, select more than one object (table/note/image), then right-click and choose **Alignment -> Align Left/Align Center/Align Right/Align Top/Align Middle/Align Bottom**.


### Change the Objects Distribution

To distribute objects on the canvas, select more than one object (table/note/image), then right-click and choose **Distribute -> Horizontal/Vertical**.

### Change Page Setup

To change paper size, orientation and margins, choose **File -> Page Setup**.

### Apply Auto Layout


To automatically arrange objects on the canvas, click  **Auto Layout** from the toolbar. To change the Auto Layout format, simply choose **Tools -> Auto Layout with** from the menu and set the following options:

Option	Description
Auto Diagram Dimension	Choose the suitable diagram dimension automatically.
Auto Size Tables to Fit	Resize the table to fit its content automatically.
Quality	The quality of the auto layout output.
Object Distance	The distance between the objects in the diagram.

## Preview and Print Model

To preview the pages before printing, simply click the  **Print Preview** button. The model can be printed to the printer or to various file formats.

### Print to a printer

Choose **File ->  Print** to send your diagram directly to the printer. You can set the printer option in the pop-up window.

## Print to a file (PDF/PNG/SVG)





Choose **File -> Print As -> PDF/PNG/SVG** to create a PDF/PNG/SVG file of your diagram.

## Reverse Engineering

**Reverse Engineering** is one of the key features of Model. This feature allows you to load already existing database structures to create new diagrams. It supports to import MySQL, PostgreSQL, Oracle, SQLite or MariaDB databases, schema, tables or views.

Navicat provides a step-by-step wizard for you to complete the task:

1. Select **Tools -> Import from Database**.
2. Select a connection.
3. Choose databases, schemas or tables you want to import.
4. Click **Start**.

You can also simply create a new model using reverse engineering in the Navicat main window. Right-click an opened database/schema, table(s) or view(s) and select  **Reverse Database to Model**,  **Reverse Schema to Model**,  **Reverse Tables to Model** or  **Reverse Views to Model** from the pop-up menu.

## Script Generation

After finishing your model, you can save table structures, views and foreign keys from the model into a script file. The **Export SQL** feature generates a SQL file for the script. Select **Tools -> Export SQL**.

### General Settings for Export SQL

#### File

Set the output file name and location.

#### Objects

Choose objects in current model you wish to export.

### Advanced Settings for Export SQL

The following options depend on the diagram database type you are chosen: MySQL, Oracle, PostgreSQL, SQLite and MariaDB.

#### Server Version

Select server version for the SQL file.

**Include Schema**

Include the schema name in file with this option is on. Otherwise, only object names are included in SQL statements.

**Default Schema**

Set the schema name for the objects without schema settings.

**Include Drop SQL**

Include drop object SQL statements in file with this option in on.

**Include Drop With CASCADE**

Include drop object SQL statements with cascade option in file with this option in on.

**Include Primary Keys**

Include primary keys in file with this option is on.

**Include Foreign Keys**

Include foreign keys in file with this option is on.

**Include Uniques**

Include uniques in file with this option is on.

**Include Indexes**

Include indexes in file with this option is on.

**Include Checks**

Include checks in file with this option is on.

**Include Excludes**

Include excludes in file with this option is on.

**Include Rules**

Include rules in file with this option is on.

**Include Triggers**

Include triggers in file with this option is on.

**Include Character Set**

Include table and field character set in file with this option is on.

**Include auto increment**

Include table auto increment values in file with this option is on.

**Include Collation**

Include table collation in file with this option is on.

# Forward Engineering

Forward engineering is one of the key features of Model. This feature allows you to compare the model with an existing database or schema, states the differences between their structures, and offer synchronizing the structures in model to the target connection.

Navicat provides a step-by-step wizard for you to complete the task.

1. Select **Tools -> Synchronize to Database**.
2. Select the synchronization type.
3. Select the source databases, schemas, tables or views and the target connection from existing connections.
4. Choose the compare and execution options.
5. Click **Compare** to generate a set of scripts that show the differences between source and target tables.
6. Select the scripts you want to run.
7. Click **Run Query**.

## Selecting Synchronization Type

### Sync with selected schemas

Set the synchronization to work on all objects in the selected schemas.

### Sync with selected objects

Set the synchronization to work on the selected objects only.

## Selecting Schemas/Objects and Connection

In this step, choose one or more schemas or objects in model to compare to the target schemas or objects. If objects in model are from existing schemas, you can select the existing schemas. Otherwise, enter a target schema name in **Objects without schema will synchronize to this schema** for the source model objects to compare to.

Then, choose target connection and database from existing connections.

## Selecting Compare and Execution Options

The following options depend on the diagram database type you are chosen: MySQL, Oracle, PostgreSQL, SQLite and MariaDB.

### Compare Identifier With Case Sensitive

Check this option if you want to compare table identifier with case sensitive option.

### Compare Tables

Check this option if you want to compare tables.



**Compare Primary Keys**

Check this option if you want to compare table primary keys.

**Compare Foreign Keys**

Check this option if you want to compare table foreign keys.

**Compare Indexes**

Check this option if you want to compare indexes.

**Compare Triggers**

Check this option if you want to compare triggers.

**Compare Character Set**

Check this option if you want to compare character set of tables.

**Compare Auto Increment Value**

Check this option if you want to compare auto increment values of tables.

**Compare Uniques**

Check this option if you want to compare uniques.

**Compare Checks**

Check this option if you want to compare checks.

**Compare Excludes**

Check this option if you want to compare excludes.

**Compare Rules**

Check this option if you want to compare rules.

**Compare Collation**

Check this option if you want to compare collation of tables.

**Compare Views**

Check this option if you want to compare views.

**Compare Definers**

Check this option if you want to compare the definers of views.

**SQL for objects to be created**

Check this option to include all related SQL statements if new database objects will be created in the target.

**SQL for objects to be changed**

Check this option to include all related SQL statements if database objects will be changed in the target.

## SQL for objects to be dropped

Check this option to include all related SQL statements if database objects will be dropped from the target.

## Continue on error

Ignore errors that are encountered during the synchronization process.

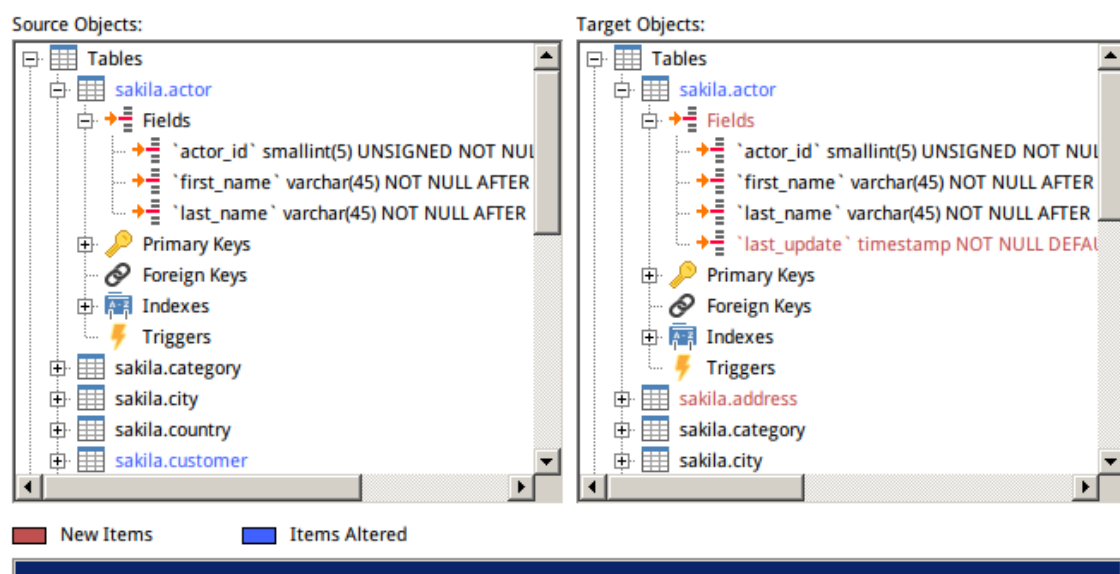
## Viewing Comparison Result

### Source Objects/Target Objects

The tree view shows the differences between model and database/schema objects after the comparison of their structures, providing with the detailed SQL statements shown in the **Queries for Modification** list.

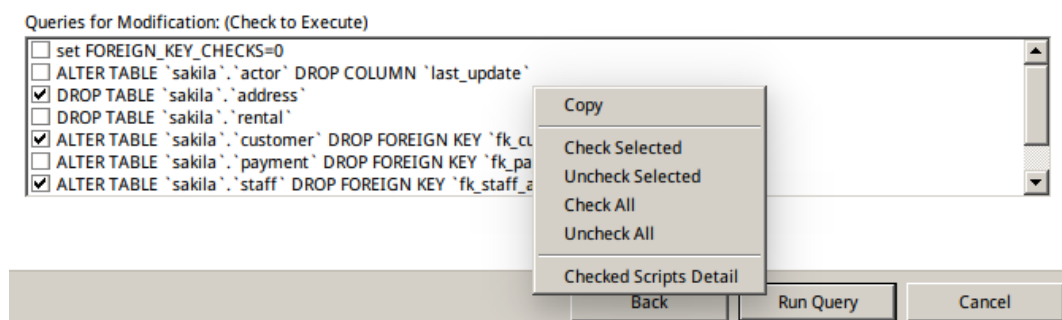
The red item represents the non-existence for the other database/schema.

The blue item represents the existence for the other database/schema, but different definition detected.



All the scripts are unchecked in the **Queries for Modification** list by default. Check the scripts you want to apply to the target.

You can highlight multiple lines of scripts, and then right-click to show the pop-up menu. Choose **Checked Scripts Detail** to view the full SQL statements you selected.



Click **Run Query** button to execute the selected query.

# Model Conversion

You can convert your models from one database type to another database type, e.g. MariaDB 10.0 physical model to PostgreSQL 9.0 physical model.



During the conversion, all data types are converted automatically. The conversion process does not change the SQL syntax of views if converting from one database type to another. If the target database version is MySQL 4.0 or below, all views will be removed.

To convert an opened model file, choose **File -> Model Conversion**. Then, select the target **Database** and **Version**.

## Model Hints and Tips

Navicat provides some useful hints to work on the model more effectively.

Action	Description
Locate Object in the Diagram Canvas	<ul style="list-style-type: none"><li>- Object selected in the Explorer's Diagram tab will be highlighted in the Diagram Canvas.</li><li>- Double-click an object in the Explorer's Diagram tab will jump to the corresponding object in the Diagram Canvas.</li></ul>
Delete Object from Model	<ul style="list-style-type: none"><li>- Select an object in the Diagram Canvas and press SHIFT+DELETE.</li></ul>
Open Table/View Designer	<ul style="list-style-type: none"><li>- Double-click a table/view in the Explorer's Model Tab or the Diagram Canvas.</li></ul>
Add table/view from Navicat Main	<ul style="list-style-type: none"><li>- Drag table/view from Navicat main window and drop to the Diagram Canvas.</li></ul>
Get Table/View Structure (SQL Statement)	<ul style="list-style-type: none"><li>- Select and copy a table/view in the Diagram Canvas, and paste it to other text editors.</li></ul>
Design Field without Table Designer	<ul style="list-style-type: none"><li>- Select and click the table name and press TAB/DOWN ARROW to add/edit fields.</li></ul> <p>Navicat will predict field types according to field names you entered.</p> <p><b>INTEGER/int/int4/NUMBER</b></p> <ul style="list-style-type: none"><li>- suffix "id", "no" (if it is the first column, it will be predicted as a primary key)</li><li>- suffix "num"</li><li>- "qty", "number"</li></ul>

	<p>- exactly "age", "count"</p> <p><b>DECIMAL(10,2)/decimal(10,2)/NUMBER/REAL/money</b></p> <p>- suffix "price", "cost", "salary"</p> <p><b>FLOAT/double/float8/NUMBER/REAL/float</b></p> <p>- "size", "height", "width", "length", "weight", "speed", "distance"</p> <p><b>DATE/datetime/date/TEXT/datetime2</b></p> <p>- "date", "time"</p> <p><b>VARCHAR(255)/varchar(255)/VARCHAR2(255)/TEXT</b></p> <p>- other field names</p> <p>Enter * before the field name to recognize as primary key. e.g. *itemNo:int. Enter : between field name and field type to custom field type, e.g. itemName:varchar(255).</p>
Reorder Field	<p>- Select a table in Diagram Canvas, then press and hold the SHIFT key. Use  to drag the field to a desired location.</p>
Delete Field	<p>- Select a table in Diagram Canvas, then press and hold the SHIFT key. Use  to drag the desired field out of the table.</p>
Add Vertex to Foreign Key/Line/Arrow	<p>- Select a foreign key/line/arrow in Diagram Canvas. Press and hold the SHIFT key and click on it to add vertex.</p>
Delete Vertex on Foreign Key/Line/Arrow	<p>- Select a foreign key/line/arrow in Diagram Canvas. Press and hold the SHIFT key and click on the vertex.</p>
Switch to Hand Mode	<p>- Press and hold the SPACE key, then move the diagram.</p>
Select a Page in Print Preview	<p>- Press and hold the SHIFT key, then point to a page to show the page number.</p> <p>- Press and hold the SHIFT key, then click a page to jump to the corresponding page in Diagram Canvas.</p>

# Advanced Tools

Navicat provides a number of powerful tools for working with data, which includes **Import Wizard**, **Export Wizard**, **Dump SQL File**, **Execute SQL File** and more.

## Import Wizard

**Import Wizard** allows you to import data to tables from CSV, TXT, XML, DBF and more. You can save your settings as a profile for setting schedule.

**Note:** Navicat Essentials version only supports to import text-based files, such as TXT, CSV, HTML, XML and JSON file.

To open the Import Wizard, click  **Import Wizard** from the table object list toolbar.

**Hint:** You can drag a supported file to the table object list or a database/schema in the Connection pane. Navicat will pop up the **Import Wizard** window. If existing table is highlighted, Navicat will import the file to the highlighted table. Otherwise, import the file to a new table.

## Setting Import File Format (Step 1)

Select one of the available import types for the source file.

## Setting Source File Name (Step 2)

Browse the source file name. The file extension in the **Import from** text box changes according to the selected import type in step 1. Select the **Encoding** for the source file.

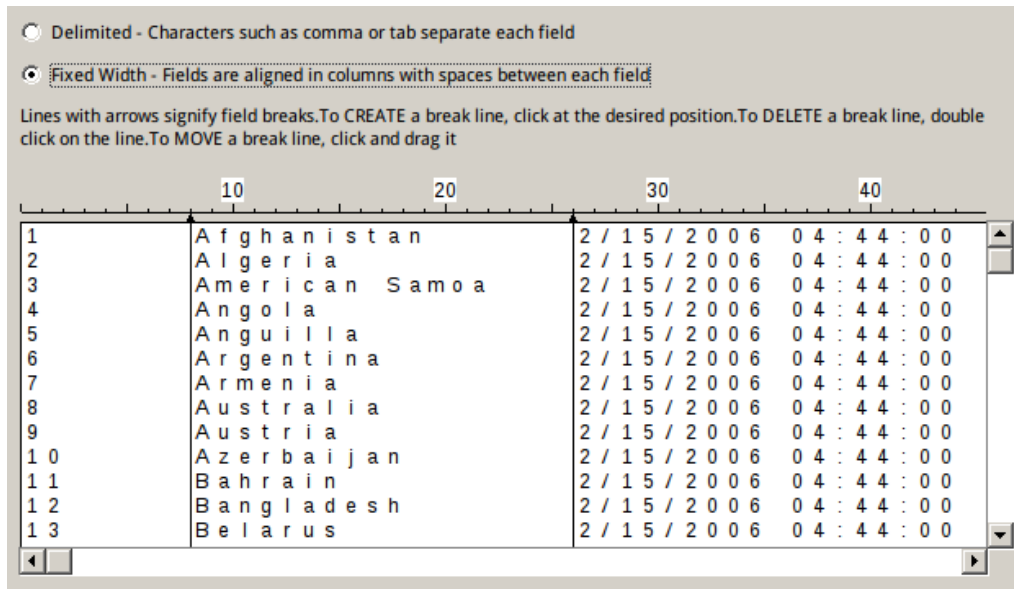
**Note:** For TXT and XML file, you can select more than one file to import.

## Setting Delimiter (Step 3) - TXT, XML

### TXT

Define **Record Delimiter**, **Field Delimiter** and **Text Qualifier** for file.

Choose **Fixed Width** to import the text file with fixed width format. To delimit the source column bounds, click on the desired position to create a break line. Simply drag it to move it or double-click it to remove it.



## XML

Define tag to identify table row.

### Consider tag attributes as table field

For example:

```
<row age="17">
<id>1</id>
<name>size</name>
</row>
```

With this option is on, Navicat will recognizes "age" as a table field together with "id" and "name", otherwise, only "id" and "name" will be imported as table fields.

**Note:** Navicat does not support multiple level of XML file.

## Setting Additional Options (Step 4) - TXT, XML, HTML

The following options depend on the file format chose in step 1.

### Field Name Row

Field name row indicates which row should Navicat recognize as Column Title.

### First Data Row

First data row indicates which row should Navicat start reading the actual data.

### Last Data Row

Last data row indicates which row should Navicat stop reading the actual data.

**Note:** If no column title are defined for the file, enter **1** for First data row and **0** for Field name row.

## Date Order, Date Delimiter, Time Delimiter, DateTime Order

Define the formats of date and time.

## Decimal Symbol

Define the format of decimal number.

## Binary Data Encoding

Set binary data is imported as **Base64** encoded or **None** encoding from file.

## Setting Target Table (Step 5)

You are allowed to define a new table name or choose to import into an existing table from the drop-down list.

**Note:** If you type a new table name in **Target Table**, the box in **New Table** will be checked automatically.

Source Table	Target Table	New Table
country	place	<input checked="" type="checkbox"/>

For importing multiple tables, all tables will be shown in the list.

Source Table	Target Table	New Table
customer	customer	<input type="checkbox"/>
country	country	<input type="checkbox"/>

## Adjusting Field Structures and Mapping Fields (Step 6)

Navicat will make assumption on the field types and length in the source table. You are allowed to choose desired type from the drop-down list.

**Hint:** For importing multiple tables, select other tables from the **Source Table** drop-down list.

Source Table:	customer				
Target Table:	customer				
	Target Field				
<input checked="" type="checkbox"/>	customer_id				
<input checked="" type="checkbox"/>	store_id				
<input checked="" type="checkbox"/>	first_name	varchar	50	0	
<input checked="" type="checkbox"/>	last_name	varchar	50	0	
<input checked="" type="checkbox"/>	email	varchar	70	0	
<input checked="" type="checkbox"/>	address_id	int	5	0	
<input checked="" type="checkbox"/>	active	varchar	2	0	
<input checked="" type="checkbox"/>	create_date	datetime	0	0	
<input checked="" type="checkbox"/>	last_update	datetime	0	0	

If you are importing data into existing table(s), you might need to map the source field names manually to the destination table or right-click and select **Smart Match All**, **Direct Match All** and **Unmatch All** from the pop-up menu for quick mapping.

Target Field	Source Field	Primary Key
customer_id		
store_id		
first_name		
last_name		
email	customer_id	
address_id	store_id	
active	first_name	
create_date	last_name	
last_update	email	
	address_id	

Target Field	Source Field	Primary Key
customer_id		
store_id		
first_name		
last_name		
email		
address_id		
active		
create_date		
last_update		

Smart Match All  
Direct Match All  
Unmatch All

## Selecting Import Mode (Step 7)

Select the import mode that defines how the data being imported.

Import mode

☒ Append: add records to the destination table

☐ Update: update record in destination with matching record from source


☐ Append/Update: if record exists in destination, update it, Otherwise, add it

☐ Delete: delete records in destination that match records in source

☐ Copy: delete all records in destination, repopulate from the source

Advanced

**Hint:** To activate the remaining options, you must enable Primary Key in step 6.

Target Field	Source Field	Primary Key
customer_id	customer_id	
store_id	store_id	
first_name	first_name	
last_name	last_name	
email	email	
address_id	address_id	
active	active	
create_date	create_date	
last_update	last_update	

Click **Advanced** button for more settings:



The following options depend on the database type you are chosen: MySQL, Oracle, PostgreSQL, SQLite and MariaDB.

### Run multiple queries in each execution

Check this option if you want to run multiple queries in each execution.

### Use extended insert statements

Insert records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');
```

### Use empty string as NULL

Import **NULL** value if the source data field contains empty string.

### Use Foreign Key constraint

Add foreign key if there is foreign key relations between tables.

### Continue on error

Ignore errors that are encountered during the import process.

### Create Auto Increment Fields

Create Auto Increment Fields during the import process.

**Note:** Support only when file type is Paradox file or DBase file.

### Import Deleted Records

Import the deleted records in the DBase file during the import process.

**Note:** Support only when file type is DBase file.

## Saving and Confirming Import (Step 8)

Click **Start** button to start the import process. You can view the running process indicating success or failure. These messages are saved in file - LogImport.txt.

**Hint:** Click **Save** button to save your settings as a profile for setting schedule.

You can click **Log** button to view the log file.

## Export Wizard

**Export Wizard** allows you to export data from tables, views, or query results to any available formats. You can save your settings as a profile for setting schedule.

**Note:** Navicat Essentials version only supports to export text-based files, such as TXT, CSV, HTML, XML and JSON file.

To open the Export Wizard, click  **Export Wizard** from the object list toolbar.

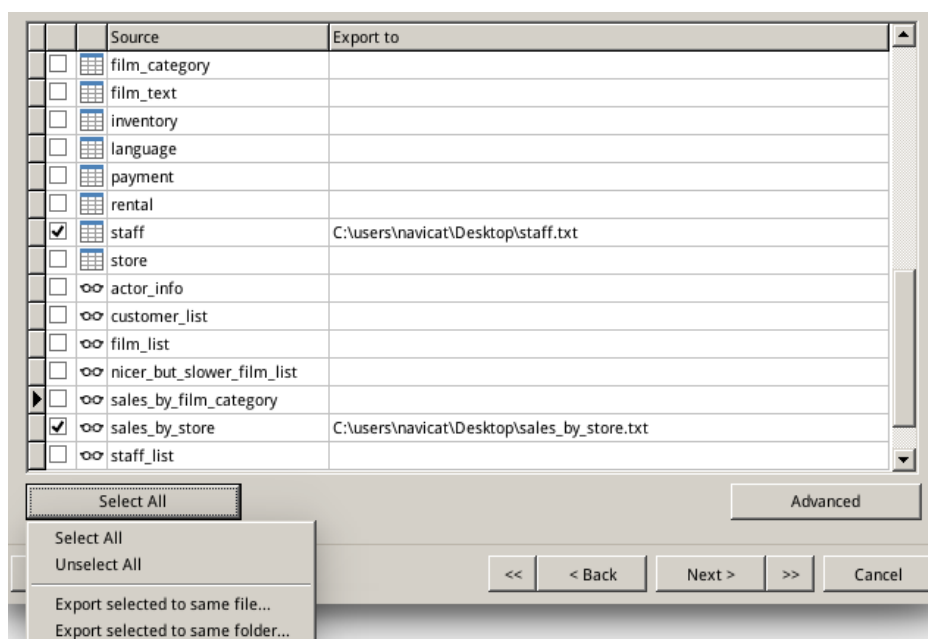
## Setting Export File Format (Step 1)

Select one of the available export formats for the target file.

## Setting Destination File Name (Step 2)

Set exported file name and location. You can check the box next to table name in **Source** directly to assign a default file name and location. If existing table is highlighted, it will be checked automatically and assigned a default file name and location. The file extension in the **Export to** text box changes according to the selected export type in step 1.

**Note:** For exporting query result, ensure the query is saved before running the Export Wizard. Otherwise, no source table displayed in here.



If you are exporting selected tables into the same target file, you can select **Export selected to same file** from **Select All** button for quick mapping.

If you are exporting selected tables into the same directory, you can select **Export selected to same folder** from **Select All** button for quick mapping.

Click **Advanced** button for more settings:

### Encoding

Select the encoding for the exported file.

### Add timestamp

Check this option if you want your file name specifies the timestamp of the export is run. Select the date/time format from the drop-down list.

## Selecting Fields for Export (Step 3)

Select table fields for export. All the fields are selected in the **Available Fields** list by default. If you want to omit some fields to be exported, uncheck the box **All Fields** first and then uncheck those fields in the **Available Fields** list.

**Note:** For exporting query result, the wizard will skip this step.

## Setting Additional Options (Step 4)

The following options depend on the file format chose in step 1.

### Include column titles

Field names will be included into the exported file if this option is on.

### Append

If you select **Export selected to same file** option for multiple tables in step 2, check this option to append records in the exported file.

### Continue on error

Ignore errors that are encountered during the export process.

### Use Attributes Format in XML

Attributes Format
<RECORDS> <RECORD OrderNo="1003" ItemNo="1" PartNo="1313" Qty="5" Discount="0"></RECORD> <RECORD OrderNo="1004" ItemNo="1" PartNo="1313" Qty="10" Discount="50"></RECORD> </RECORDS>
Non-Attributes Format
<RECORDS> <RECORD> <OrderNo>1003</OrderNo> <ItemNo>1</ItemNo> <PartNo>1313</PartNo> <Qty>5</Qty> <Discount>0</Discount> </RECORD> <RECORD>

```
<OrderNo>1004</OrderNo>
<ItemNo>1</ItemNo>
<PartNo>1313</PartNo>
<Qty>10</Qty>
<Discount>50</Discount>
</RECORD>
</RECORDS>
```

Define **Record Delimiter**, **Field Delimiter** and **Text Qualifier** for file.

#### **Date Order, Date Delimiter, Zero Padding Date, Time Delimiter**

Define the formats of date and time.

#### **Decimal Symbol**

Define the format of decimal number.

#### **Binary Data Encoding**

Set binary data is exported as **Base64** encoded or **None** encoding in file.


## Saving and Confirming Export (Step 5)

Click **Start** button to start the export process. You can view the running process indicating success or failure. These messages are saved in file - LogExport.txt.

**Hint:** Click **Save** button to save your settings as a profile for setting schedule.

You can click **Open** button to open the log file.

## Data Transfer (Available only in Full Version)

Navicat allows you to transfer database objects from one database and/or schema to another, or to a sql file. The target database and/or schema can be on the same server as the source or on another server. You can save your settings as a profile for setting schedule. Select **Tools** ->  **Data Transfer** from the main menu.

**Hint:** You can drag tables to a database/schema in the Connection pane. If the target database/schema is within the same connection, Navicat will copy the table directly. Otherwise, Navicat will pop up the **Data Transfer** window.


To open a saved profile, select the profile and click the **Load** button or double-click it in the **Profiles** tab.


## General Settings for Data Transfer

### **Source**

Define connection, database and/or schema for the source.

All the database objects are selected in the **Database Objects** list by default. If you do not want some database objects to be transferred, uncheck them.

 With this option is on, only the checked database objects will be transferred. However, if you add any new database objects in the source database and/or schema after you create your data transfer profile, the newly added database objects will not be transferred unless you manually modify the **Database Objects** list.

 Choose this option if you wish all the database objects being transferred to the target database/schema, all newly added database objects will also be transferred without amending the data transfer profile.

## Target

### Connection

Transfer your selected database objects directly to a connection, database and/or schema.

### File

Transfer your selected database objects directly to a text file. You can select different **SQL Format** and **Encoding** for the file.

## Advanced Settings for Same Server Type Data Transfer

In this tab, you can choose the advanced settings for transferring between same server type or between MySQL and MariaDB.

The following options depend on the database type you are chosen: MySQL, Oracle, PostgreSQL, SQLite and MariaDB.

### Create tables

Create tables in the target database and/or schema with this option is on.

Suppose this option is unchecked and tables already exist in the target database/schema, then all data will be appended to the destination tables.

### Include indexes

Include indexes in the table with this option is on.

### Include foreign key constraints

Include foreign keys in the table with this option is on.

### Include engine/table type

Include table type with this option is on.

### Include character set

Include character set in the table with this option is on.

**Include auto increment**

Include auto increment in the table with this option is on.

**Include other table options**

Include other options in the table with this option is on.

**Include unique constraints**

Include uniques in the table with this option is on.

**Include rules**

Include rules in the table with this option is on.

**Include check constraints**

Include checks in the table with this option is on.

**Include triggers**

Include triggers in the table with this option is on.

**Include excludes**

Include exclusion constraints in the table with this option is on.

**Convert object name to**

Check this option if you require convert object names to **Lower case** or **Upper case** during the process.

**Insert records**

Check this option if you require all records to be transferred to the destination database and/or schema.

**Lock target tables**

Lock the tables in the target database and/or schema during the data transfer process.

**Use transaction**

Check this option if you use transaction during the data transfer process.

**Use complete insert statements**

Insert records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindy', '23');  
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');  
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');
```

**Use extended insert statements**

Insert records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');
```

### Use delayed insert statements

Insert records using *DELAYED* insert SQL statements.

Example:

```
INSERT DELAYED INTO `users` VALUES ('1', 'Peter McKindsy', '23');
```

```
INSERT DELAYED INTO `users` VALUES ('2', 'Johnson Ryne', '56');
```

```
INSERT DELAYED INTO `users` VALUES ('0', 'katherine', '23');
```

### Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

### Use hexadecimal format for BLOB

Insert BLOB data as hexadecimal format.

### Continue on error

Ignore errors that are encountered during the transfer process.

### Lock source tables

Lock the tables in the source database and/or schema so that any update on the table is not allowed once the data transfer is triggered off.

### Drop target objects before create

Check this option if database objects already exist in the target database and/or schema, the existing objects will be deleted once the data transfer starts.

### Create target database/schema if not exist

Create a new database/schema if the database/schema specified in target server does not exist.

### Use DDL from SHOW CREATE TABLE

If this option is on, DDL will be used from show create table.

### Use DDL from sqlite\_master

If this option is on, DDL will be used from the *SQLITE\_MASTER* table.

## Advanced Settings for Cross Server Data Transfer (Available only in Navicat Premium)

Navicat Premium supports transferring table with data across different server types, e.g. from MySQL to Oracle. If you are transferring between MySQL and MariaDB, you can refer to [Advanced Settings for Same Server Type Data Transfer](#).

The following options depend on the database type you are chosen: MySQL, Oracle, PostgreSQL, SQLite and MariaDB.

### Create tables

Create tables in the target database and/or schema with this option is on.

Suppose this option is unchecked and tables already exist in the target database/schema, then all data will be appended to the destination tables.

### Include indexes

Include indexes in the table with this option is on.

### Include foreign key constraints

Include foreign keys in the table with this option is on.

### Convert object name to

Check this option if you require convert object names to **Lower case** or **Upper case** during the process.

### Insert records

Check this option if you require all records to be transferred to the destination database and/or schema.

### Lock target tables

Lock the tables in the target database and/or schema during the data transfer process.

### Use transaction

Check this option if you use transaction during the data transfer process.

### Use complete insert statements

Insert records using complete insert syntax.

Example:

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('1', 'Peter McKindy', '23');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('2', 'Johnson Ryne', '56');
```

```
INSERT INTO `users` (`ID Number`, `User Name`, `User Age`) VALUES ('0', 'katherine', '23');
```

### Use extended insert statements

Insert records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');
```

### Use delayed insert statements

Insert records using *DELAYED* insert SQL statements.

Example:

```
INSERT DELAYED INTO `users` VALUES ('1', 'Peter McKindy', '23');
```

```
INSERT DELAYED INTO `users` VALUES ('2', 'Johnson Ryne', '56');
```

```
INSERT DELAYED INTO `users` VALUES ('0', 'katherine', '23');
```



### Run multiple insert statements

Check this option if you want to run multiple insert statements in each execution, which will make the data transfer process faster.

### Use hexadecimal format for BLOB

Insert BLOB data as hexadecimal format.

### Continue on error

Ignore errors that are encountered during the transfer process.

### Lock source tables

Lock the tables in the source database and/or schema so that any update on the table is not allowed once the data transfer is triggered off.

### Drop target objects before create

Check this option if database objects already exist in the target database and/or schema, the existing objects will be deleted once the data transfer starts.

### Create target database/schema if not exist

Create a new database/schema if the database/schema specified in target server does not exist.

## Data Synchronization (Available only in Full Version)

Navicat allows you to transfer data from one database and/or schema to another with detailed analytical process. In other words, Navicat provides the ability for data in different databases and/or schemas to be kept up-to-date so that each repository contains the same information. You are not only authorized to rollback the transferring process, but also insert, delete and update records to the destination. You can save your settings as a profile for setting schedule.

Select **Tools** ->  **Data Synchronization** from the main menu.

All tables must contain primary keys and all table structures must be identical between the source and target. You could apply Structure Synchronization before Data Synchronization.

The messages indicate the running process success or failure are saved in file - LogSynchronize.txt.

To open a saved profile, select the profile and click **Load** button or double-click it in **Profiles** tab.

**Note:** For Oracle server, BLOB, CLOB, NCLOB, LONG and LONG RAW data are skipped during the data synchronization process. TIMESTAMP primary key cannot synchronize (insert, update) with Database Link to 9i server. RAW primary key cannot synchronize (insert, update, delete) with Database Link to any server, without error.

Navicat Premium and Navicat for MySQL support synchronize between MySQL and MariaDB.

## General Settings for Data Synchronization

### Source/Target

Define connection, database and/or schema for the source and target.

**Note:** For Oracle server, you need to create [Public/Private](#) Database Link to the target Oracle database before.

### Source Table/Target Table

Only tables which contain identical table names between the source and target are mapped in the list by default. If you do not want some tables to be synchronized, disable them manually from the drop-down list.

**Hint:** You can preview the outcome before execution.

## Advanced Settings for Data Synchronization

### Use Transaction

Rollback all data when error occurs.

### Show synchronization detail


Check this option if you want to list the details process in **Message Log** tab during the synchronization.

**Note:** The process will be faster if this option is unchecked.

### Insert records, Delete records, Update records

Check these options to performing such actions to the target when data are synchronized.

## Structure Synchronization (Available only in Full Version)

Navicat allows you to compare and modify the table structures with detailed analytical process. In other words, Navicat compares tables between two databases and/or schemas and states the differential in structure. Select **Tools** ->  **Structure Synchronization** from the main menu.

To open a saved profile, select the profile and click **Load** button or double-click it in **Profiles** tab.

**Note:** Available only for MySQL, Oracle, PostgreSQL and MariaDB. Navicat Premium and Navicat for MySQL support synchronize between MySQL and MariaDB.

## General Settings for Structure Synchronization

The following options depend on the database type you are chosen: MySQL, Oracle, PostgreSQL and MariaDB.

### Source/Target

Define connection, database and/or schema for the source and target.

**Compare Tables**

Check this option if you want to compare tables between the source and target.

**Compare Primary Keys**

Check this option if you want to compare table primary keys.

**Compare Foreign Keys**

Check this option if you want to compare table foreign keys.

**Compare Character Set**

Check this option if you want to compare character set of tables.

**Compare Auto Increment Value**

Check this option if you want to compare auto increment values of tables.

**Compare Partitions**

Check this option if you want to compare table partitions.

**Compare Uniques**

Check this option if you want to compare uniques.

**Compare Checks**

Check this option if you want to compare checks.

**Compare Excludes**

Check this option if you want to compare exclude constraints.

**Compare Views**

Check this option if you want to compare views.

**Compare Functions**

Check this option if you want to compare functions.

**Compare Events**

Check this option if you want to compare events.

**Compare Definer**

Check this option if you want to compare definer.

**Compare Indexes**

Check this option if you want to compare indexes.

**Compare Sequences**

Check this option if you want to compare sequences.

### **Compare Triggers**

Check this option if you want to compare triggers.

### **Compare Rules**

Check this option if you want to compare rules.

### **Compare Tablespace and Physical Attributes**

Check this option if you want to compare tablespace and physical attributes.

### **SQL for objects to be created**

Check this option to include all related SQL statements if new database objects will be created in the target.

### **SQL for objects to be changed**

Check this option to include all related SQL statements if database objects will be changed in the target.

### **SQL for objects to be dropped**

Check this option to include all related SQL statements if database objects will be dropped from the target.

### **Drop with CASCADE**

Check this option if you want to cascade to drop the dependent database objects.

### **Compare after execution**

Compare tables after the synchronization is executed.

### **Continue on error**

Ignore errors that are encountered during the synchronization process.

### **Create inheriting parent**

Create tables of inheriting parents during the synchronization process.

## **Structure Synchronization Result**

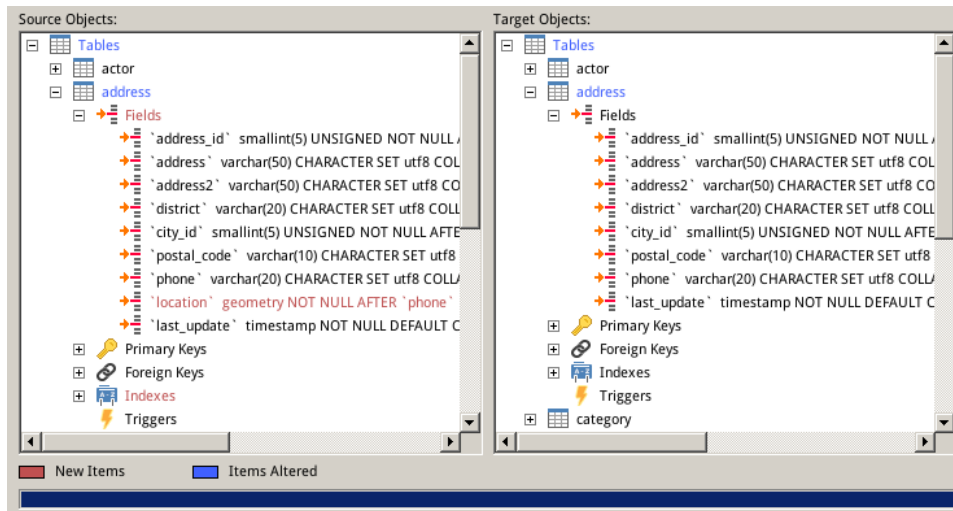
### **Source Objects/Target Objects**

The tree view shows the differences between the source and target database and/or schema after the comparison of their structures, providing with detailed SQL statements shown in the **Queries for Modification** list.

The red item represents the non-existence for the other database/schema.

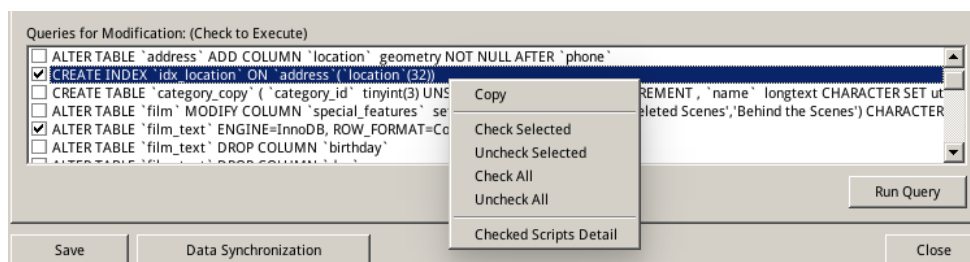
The blue item represents the existence for the other database/schema, but different definition detected.

You are allowed to edit object structure manually, right-click the object in the tree view and select **Edit** from the pop-up menu to open the relevant designer.



All the scripts are unchecked in the **Queries for Modification** list by default. Check the scripts you want to apply to the target.


You can highlight multiple lines of scripts, and then right-click to show the pop-up menu. Choose **Checked Scripts Detail** to view the full SQL statements you selected.



Click **Run Query** button to execute the selected query.

## Backup/Restore (Available only in Full Version)

A secure and reliable server is closely related to performing regular backups, as failures will probably occur sometimes - caused by attacks, hardware failure, human error, power outages, etc.

Navicat allows you to backup/restore database objects for your database. You can save your settings as a profile for setting schedule. Click  to open an object list for **Backup**.

**Hint:** Backup files are stored under [Settings Location](#).

**Note:** Available only for MySQL, PostgreSQL, SQLite and MariaDB. To backup Oracle, see [Oracle Data Pump](#).

## Backup

### General Properties

You can enter comment for the backup file.

## Object Selection

Choose database objects you wish to backup.

## Advanced Properties

The following options depend on the database type you are chosen: MySQL, PostgreSQL, SQLite and MariaDB.

### Compressed

Check this option if you want to produce smaller backup file.

To compress or decompress the backup file in the Object List pane, right-click the backup file and select **Compress Backup** or **Decompress Backup** from the pop-up menu.

**Hint:** compressed (.psc), uncompressed (.psb).

### Lock All Tables

Lock all objects while backup is being processed.

### Use Single Transaction (InnoDB only)

If a table uses InnoDB storage engine, with this option is on, Navicat uses transaction before the backup process starts.


### Use specify file name

Define your file name for backup. Otherwise, your backup file will be named as "2007-05-10 17:38:20" for example.

## Restore

**Restore** feature will firstly drop the selected objects of the database, then recreate the new objects according to your backup. Finally, inserting the data.

To restore a backup to an existing database, open a database and select an existing backup file. Click  **Restore Backup** from the backup object list toolbar.

To restore a backup to a new database, create and open a new database and click  **Restore Backup** from the backup object list toolbar. Browse the backup file.

**Note:** You must have Create, Drop and Insert privileges ([MySQL/MariaDB](#) or [PostgreSQL](#)) to run the restore.

## General Properties

Show information of the backup file.

## Object Selection

Choose database objects you wish to restore.

## Advanced Properties

The following options depend on the database type you are chosen: MySQL, PostgreSQL, SQLite and MariaDB.

### Use Transaction

Rollback all data when error occurs.

### Continue on error

Ignore errors that are encountered during the restore process.

### Lock tables for write

Lock the tables to prevent user to modify tables during the restore process.

### Use extended insert statements

Insert records using extended insert syntax.

Example:

```
INSERT INTO `users` VALUES ('1', 'Peter McKindy', '23'), ('2', 'Johnson Ryne', '56'), ('0', 'Katherine', '23');
```

### Run multiple queries in each execution

Check this option if you want to run multiple queries in each execution, which will make the restore process faster.

### Create tables

Create tables during the restore process with this option is on.

### Create records

Restore table records with this option is on. Otherwise, only table structures will be restored.

### Create indexes

Create indexes for the restored table with this option is on.

### Create triggers

Create triggers for the restored table with this option is on.

### Overwrite existing tables

Overwrite if tables already exist in the database/schema.

### Overwrite existing views

Overwrite if views already exist in the database/schema.

### Overwrite existing functions

Overwrite if functions already exist in the database/schema.

### Overwrite existing events

Overwrite if events already exist in the database/schema.

### Overwrite existing sequences

Overwrite if sequences already exist in the database/schema.

### Overwrite existing indexes

Overwrite if indexes already exist in the database/schema.

### Overwrite existing triggers


Overwrite if triggers already exist in the database/schema.

### Insert Auto Increment Values


Insert auto increment values in the database/schema.

## Extract SQL

**Extract SQL** allows extracting SQL into a SQL file from your backup file.

To extract SQL from a backup file, open a database and select an existing backup file. Click  **Extract SQL** from the backup object list toolbar.

## Batch Job/Schedule (Available only in Full Version)

Navicat allows you to create a batch job for setting schedule to execute at one or more regular intervals, beginning and ending at a specific date and time. Batch job can be created for Query, Backup, Data Transfer, Data Synchronization, Import and Export from databases. You can define a list of actions to be performed within one batch job, either run it manually or at the specified time/periodically. Click  to open an object list for **Schedule**.

The messages indicate the running process success or failure are saved in file - LogCmd.txt.

Click  **Set Task Schedule** to set schedule for batch job.

**Note:** Please save the batch job before setting schedule. Passwords must be saved in [Connection Properties](#) before running your schedule.

Connection Properties	
User Name:	<input type="text" value="root"/>
Password:	<input type="password" value="*****"/>
<input checked="" type="checkbox"/> Save Password	

### Setup Schedule

The pattern follows a crontab entry's format. If a field is left without a value, then all the values will be used. For example, if the "weekday" field is empty, then the system will treat the field to be entered with "0, 1, 2, 3, 4, 5, 6". Use commas to separate values. For example, "0, 1, 3, 6". Use hyphen, without spaces to indicate values. For example, "0-4".



Example: The batch job will be executed at 6:30pm every weekday.

**Tips:** What is crontab?

For commands that need to be executed repeatedly (e.g., hourly, daily, or weekly), you can use the crontab command. The crontab command creates a crontab file containing commands and instructions for the cron daemon to execute. You can use the crontab command with the following options:

crontab -a filename	Install filename as your crontab file. On many systems, this command is executed simply as crontab filename (i.e., without the -a option).
crontab -e	Edit your crontab file, or create one if it doesn't already exist.
crontab -l	Display your crontab file.
crontab -r	Remove your crontab file.
crontab -v	Display the last time you edited your crontab file. (This option is only available on a few systems.)
crontab -u user	Used in conjunction with other options, this option allows you to modify or view the crontab file of user. When available, only administrators can use this option.

Each entry in a crontab file consists of six fields, specifying in the following order:


minute(s) hour(s) day(s) month(s) weekday(s) command(s)



The fields are separated by spaces or tabs. The first five are integer patterns and the sixth is the command to execute.

Field	Value	Description
minute	0-59	The exact minute that the command sequence executes
hour	0-23	The hour of the day that the command sequence executes
day	1-31	The day of the month that the command sequence executes
month	1-12	The month of the year that the command sequence executes
weekday	0-6	The day of the week that the command sequence executes (Sunday = 0, Monday = 1, Tuesday = 2, and so forth)
command	Special	The complete sequence of commands to execute. The command string must conform



		to Bourne <a href="#">shell</a> syntax. Commands, executables (such as scripts), or combinations are acceptable.
--	--	------------------------------------------------------------------------------------------------------------------

## General Settings for Batch Job/Schedule

Move objects from the **Available Jobs** list to the **Selected Jobs** list by using  **Select** button or double-clicking. To delete the objects from the selected jobs list, remove them in the same way. You are allowed to run profiles from different servers in a single batch job/schedule.

To rearrange the sequence of the selected jobs, use  **Move Up** or  **Move Down** buttons.

To backup whole server, you can select the connection and choose **Backup Server xxx**. (To backup your connection settings, see [Migrate Navicat to new computer](#).)

Exported file can be added to the batch job as mail attachment. Select the job in the **Selected Jobs** list and click  **Add Attachment** or  **Remove Attachment** to add or remove the mail attachment.

To find Data Transfer or Data Synchronization profile, choose  **Navicat** at the top on the left panel.

## Advanced Settings for Batch Job/Schedule

### Send Email

Navicat allows you to generate and send personalized e-mails with results returned from a schedule. The resultset(s) can be emailed to multiple recipients. Check this option and enter required information.

#### From

Specify e-mail address of sender. For example, someone@navicat.com.

#### To, CC

Specify e-mail addresses of each recipient, separating them with a comma or a semicolon (;).

#### Subject

Specify the email subject with customized format.

#### Body

Write email content.

#### Host (SMTP Server)

Enter your Simple Mail Transfer Protocol (SMTP) server for outgoing messages.

#### Port

Enter the port number you connect to your outgoing e-mail (SMTP) server. Default value is **25**.

#### Use Authentication

Check this option and enter **User Name** and **Password** if your SMTP server requires authorization to send email.

## Secure Connection

Specify the connection to use **TLS**, **SSL** secure connection or **Never**.

## Send test mail

Navicat will send you a test mail indicating success or failure.

## Batch Job Converter (Available only in Navicat Premium)

Navicat Premium allows you to convert saved batch jobs from either Navicat for MySQL, Navicat for Oracle, Navicat for PostgreSQL, Navicat for SQLite and Navicat for MariaDB to it. Right-click in the Object List pane and select **Batch Job Converter** from the pop-up menu.

### Delete original batch jobs

Check this option if you want to delete the original batch jobs in Navicat. If the original batch job is deleted, the scheduled batch job will not work until it is set again in Navicat Premium or the original application.

### Overwrite existing batch jobs

Check this option if you want to overwrite the existing batch jobs in Navicat Premium.

### Append when batch job exists

Check this option and enter the name of existing batch job if you want to append the details to the existing batch jobs in Navicat Premium.

Click **Start** button to start the import process.

## Dump SQL File/Execute SQL File




Navicat allows you to backup and restore your database/schema/table(s) using the **Dump SQL File** and **Execute SQL File** features.

To backup your database/schema/table(s), right-click it and select **Dump SQL File -> Structure And Data** or **Structure Only** from the pop-up menu.


To restore your database/schema/table(s) or execute SQL file, right-click and select **Execute SQL File** from the pop-up menu.

**Hint:** You can drag a .sql file to the table object list or a database/schema in the Connection pane. Navicat will pop up the **Execute SQL File** window.

## Print Database/Schema/Table Structure (Available only in Full Version)

Navicat allows you to view and print database, schema and table structures. Right-click the database/schema/table(s) and select  **Print Database** or  **Print Schema** or  **Print Tables** from the pop-up menu.

# Server Security

Navicat provides security management tool for your server. You can add, edit, delete users, grant/revoke privileges on the selected database and their database objects. Click  to open an object list for **User**. The Object List pane displays all users that exist in the server.

## MySQL/MariaDB Security

Information about user privileges is stored in the **user**, **db**, **host**, **tables\_priv**, **columns\_priv**, and **procs\_priv** tables in the **mysql** database (that is, in the database named mysql). The MySQL server reads the contents of these tables when it starts.

MySQL access control involves two stages when you run a client program that connects to the server:

Stage 1: The server checks whether it should allow you to connect.

Stage 2: Assuming that you can connect, the server checks each statement you issue to determine whether you have sufficient privileges to perform it. For examples: Create table privilege, Drop table privilege or Alter table privilege.

The server uses the **user**, **db**, and **host** tables in the **mysql** database at both stages of access control.

## MySQL/MariaDB User Designer

### General Properties

#### User name

Set name for user.

#### Host

A host name where the database is situated or the IP address of the server.

#### Password

Set **Password** and re-type it in the **Confirm Password** text box.

### Advanced Properties

#### Max queries per hour, Max updates per hour, Max connections per hour

These options limit the number of queries, updates, and logins a user can perform during any given one-hour period. If they are set as 0 (the default), this means that there is no limitation for that user.

#### Max user connections

This option limits the maximum number of simultaneous connections that the account can make. If it is set as 0 (the

default), the *max\_user\_connections* system variable determines the number of simultaneous connections for the account.

### Use OLD\_PASSWORD encryption

The password hashing mechanism was updated in MySQL 4.1 to provide better security and to reduce the risk of passwords being intercepted. However, this new mechanism is understood only by MySQL 4.1 (and newer) servers and clients, which can result in some compatibility problems. A 4.1 or newer client can connect to a pre-4.1 server, because the client understands both the old and new password hashing mechanisms. However, a pre-4.1 client that attempts to connect to a 4.1 or newer server may run into difficulties.

Enable this option if you wish to maintain backward compatibility with pre-4.1 clients under circumstances where the server would otherwise generate long password hashes. The option does not affect authentication (4.1 and later clients can still use accounts that have long password hashes), but it does prevent creation of a long password hash in the *user* table as the result of a password-changing operation.

### SSL Type

MySQL can check X509 certificate attributes in addition to the usual authentication that is based on the username and password. To specify SSL-related options for a MySQL account, use the *REQUIRE* clause of the *GRANT* statement.

ANY	This option tells the server to allow only SSL-encrypted connections for the account.
X509	This means that the client must have a valid certificate but that the exact certificate, issuer, and subject do not matter. The only requirement is that it should be possible to verify its signature with one of the CA certificates.
SPECIFIED	<p><b>Issuer</b></p> <p>This places the restriction on connection attempts that the client must present a valid X509 certificate issued by CA <i>issuer</i>. If the client presents a certificate that is valid but has a different issuer, the server rejects the connection. Use of X509 certificates always implies encryption, so the SSL option is unnecessary in this case.</p> <p><b>Subject</b></p> <p>This places the restriction on connection attempts that the client must present a valid X509 certificate containing the subject <i>subject</i>. If the client presents a certificate that is valid but has a different subject, the server rejects the connection.</p> <p><b>Cipher</b></p> <p>This is needed to ensure that ciphers and key lengths of sufficient strength are used. SSL itself can be weak if old algorithms using short encryption keys are used. Using this option, you can ask that a specific cipher method is used to allow a connection.</p>

### Server Privileges

In the grid, check **Granted** option against the server privilege listed in **Privilege** to assign this user to have that privilege. Multiple privileges can be granted.

## Privileges

To edit specific object privileges for user, click  **Add Privilege** to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **State** option against the privilege listed in **Privilege** to assign this user to have that privilege.  
Multiple privileges can be granted.

## Oracle Security

Oracle manages database access permissions using users and roles. Users own schema objects (for example, tables, views) and can assign privileges on those objects to other users to control who has access to which objects.

In addition to the user accounts that you create, the database includes a number of user accounts that are automatically created upon installation. Administrative accounts: **SYS**, **SYSTEM**, **SYSMAN**, and **DBSNMP**. Administrative accounts are highly privileged accounts to perform administrative tasks such as starting and stopping the database, managing database memory and storage, creating and managing database users, and so on. Your database may also include sample schemas (**SCOTT**, **HR**, **OE**, **OC**, **PM**, **IX** and **SH**), which are a set of interlinked schemas that enable Oracle documentation and Oracle instructional materials to illustrate common database tasks.

When you create a database object, you become its owner. By default, only the owner of an object can do anything with the object. In order to allow other users to use it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Ordinarily, only the object's owner (or a superuser) can grant or revoke privileges on an object. However, it is possible to grant a privilege **Admin Option/Grant Option**, which gives the recipient the right to grant it in turn to others. If the grant option is subsequently revoked then all who received the privilege from that recipient (directly or through a chain of grants) will lose the privilege.

**Note:** The special name **PUBLIC** is accessible to every database user, all privileges and roles granted to **PUBLIC** are accessible to every database user.

## Oracle User Designer

### General Properties

#### User name

Set name for user.

#### Authentication

Select the authentication method.

Password	<b>Password</b>
----------	-----------------

	<p>Set <b>Password</b> and re-type it in the <b>Confirm Password</b> text box.</p> <p><b>Expire Password</b></p> <p>Expire the user's password. This setting forces the user or the DBA to change the password before the user can log in to the database.</p>
External	An external user must be authenticated by an external service, such as an operating system or a third-party service.
Global	<p>A global user must be authorized by the enterprise directory service (Oracle Internet Directory).</p> <p><b>X.500 Name</b></p> <p>Enter the X.509 name at the enterprise directory service that identifies this user.</p>

### Default Table Space

Choose the default tablespace for objects that the user creates.

### Temporary Table Space

Choose the tablespace or tablespace group for the user's temporary segments.

### Profile

Choose the profile that assign to the user.

### Lock Account

Lock the user's account and disable access.

### Member of

In the grid, check **Granted**, **Admin Option** or **As Default** option against the role listed in **Role Name** to assign this user to be a member of selected role. Multiple roles can be granted.

### Quotas

In the grid, specify the maximum amount of space the user can allocate in the tablespaces. Enter the **Quota** and choose the **Unit** of the **Tablespace**. **Unlimited** lets the user allocate space in the tablespace without bound. Multiple tablespaces can be set.

### Server Privileges

In the grid, check **Granted** or **Admin Option** option against the server privilege listed in **Privilege** to assign this user to have that privilege. Multiple privileges can be granted.

### Privileges

To edit specific object privileges for user, click  **Add Privilege** to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.



3. In the grid, check **Granted** or **Grant Option** option against the privilege listed in **Privilege** to assign this user to have that privilege. Multiple privileges can be granted.

## Oracle Role Designer

### General Properties

#### Role name

Set name for role.

#### Authentication

Select the authentication method.

Password	<b>Password</b> Set <b>Password</b> and re-type it in the <b>Confirm Password</b> text box.
External	An external user must be authenticated by an external service, such as an operating system or a third-party service, before enabling the role.
Global	A global user must be authorized to use the role by the enterprise directory service before the role is enabled at login.
Not Identified	The role is authorized by the database and that no password is required to enable the role.

### Member of

In the grid, check **Granted** or **Admin Option** option against the role listed in **Role Name** to assign this role to be a member of selected role. Multiple roles can be granted.

### Members

In the grid, check **Granted** or **Admin Option** option against user listed in **Member** to assign the selected user to be a member of this role. Multiple users can be granted.

### Server Privileges

In the grid, check **Granted** or **Admin Option** option against the server privilege listed in **Privilege** to assign this role to have that privilege. Multiple privileges can be granted.

### Privileges

To edit specific object privileges for role, click  **Add Privilege** to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant** option against the privilege listed in **Privilege** to assign this role to have that privilege. Multiple privileges can be granted.

# PostgreSQL Security

PostgreSQL manages database access permissions using users and groups. Users own database objects (for example, tables) and can assign privileges on those objects to other users to control who has access to which objects.

**Note:** Starting from PostgreSQL version 8.1, users and groups were no longer distinct kinds of entities, now there are only roles. Any role can act as a user, a group, or both. The concept of roles subsumes the concepts of users and groups.

Only a superuser (a user who is allowed all rights) can add/delete users. PostgreSQL installs a single superuser by default named **postgres**. All other users must be added by this user, or by another subsequently added superuser.

When you create a database object, you become its owner. By default, only the owner of an object can do anything with the object. In order to allow other users to use it, privileges must be granted. (However, users that have the superuser attribute can always access any object.)

Ordinarily, only the object's owner (or a superuser) can grant or revoke privileges on an object. However, it is possible to grant a privilege **With Grant Option**, which gives the recipient the right to grant it in turn to others. If the grant option is subsequently revoked then all who received the privilege from that recipient (directly or through a chain of grants) will lose the privilege.

**Note:** The special name **public** can be used to grant a privilege to every role (user/group) on the system.

## PostgreSQL Server 7.3 to 8.0

PostgreSQL version 7.3 to 8.0 manages database access permissions using **Users** and **Groups**.

## PostgreSQL User Designer

### General Properties

#### User name

Set name for user.

#### User ID

Specify an ID for the user. This is normally not necessary, but may be useful if you need to recreate the owner of an orphaned object. If this is not specified, the highest assigned user ID plus one (with a minimum of 100) will be used as default.

#### Password

Set **Password** and re-type it in the **Confirm Password** text box.

**Note:** If you do not plan to use password authentication you can omit this option, but then the user will not be able to connect if you decide to switch to password authentication.

## Password Encryption

This option control whether the password is stored **ENCRYPTED** or **UNENCRYPTED** in the system catalogs. (If neither is specified, the default behavior is determined by the configuration parameter *password\_encryption*.)

## Expiry Date

Set a date and time after which the user's password is no longer valid. If this clause is omitted the password will be valid for all time.

## Superuser

Check this option to define the user as a superuser.

## Can create database

Check this option to define the user to be allowed to create databases.

## Member of

In the grid, check **Granted** option against the group listed in **Group Name** to assign this user to be a member of selected group. Multiple groups can be granted.

## Privileges

To edit specific object privileges for user, click  **Add Privilege** to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Granted** or **Grant Option** option against the privilege listed in **Privilege** to assign this user to have that privilege. Multiple privileges can be granted.

# PostgreSQL Group Designer

## General Properties

### Group name

Set name for group.

### Group ID

Specify an ID for the group. This is normally not necessary, but may be useful if you need to recreate a group referenced in the permissions of some object. If this is not specified, the highest assigned group ID plus one (with a minimum of 100) will be used as default.

## Members

In the grid, check **Granted** option against the user listed in **Member** to assign selected user to be a member of this group. Multiple users can be granted.

## Privileges

To edit specific object privileges for group, click  **Add Privilege** to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Grant** option against the privilege listed in **Privilege** to assign this group to have that privilege. Multiple privileges can be granted.

## PostgreSQL Server 8.1 or above

Starting from PostgreSQL version 8.1, users and groups were no longer distinct kinds of entities, now there are only **Roles**. Any role can act as a user, a group, or both. The concept of roles subsumes the concepts of users and groups.

## PostgreSQL Role Designer

### General Properties

#### Role Name

Set name for role.

#### Role ID

Specify an ID for the role. This is normally not necessary, but may be useful if you need to recreate the owner of an orphaned object. If this is not specified, the highest assigned role ID plus one (with a minimum of 100) will be used as default.

**Note:** In PostgreSQL versions 8.1 or above, the specified ID will be ignored, but is accepted for backwards compatibility.

#### Can login

Check this option to create a role that allow to login. A role having this option can be thought of as a user. Roles without this attribute are useful for managing database privileges, but are not users in the usual sense of the word.

#### Password

Set **Password** and re-type it in the **Confirm Password** text box.

**Note:** If you do not plan to use password authentication you can omit this option, but then the role will not be able to connect if you decide to switch to password authentication.

#### Password Encryption

This option control whether the password is stored **ENCRYPTED** or **UNENCRYPTED** in the system catalogs. (If neither is specified, the default behavior is determined by the configuration parameter *password\_encryption*.)

#### Connection Limit

If role can log in, this specifies how many concurrent connections the role can make. -1 (the default) means no limit.

### Expiry Date

Set a date and time after which the role's password is no longer valid. If this clause is omitted the password will be valid for all time.

### Superuser

Check this option to determine the new role is a superuser, who can override all access restrictions within the database.

### Can create database

Check this option to define a role's ability to create databases.

### Can create role

Check this option to allow creating roles.

### Inherit privileges

Check this option to determine whether a role inherits the privileges of roles it is a member of.

### Can update system catalog

Check this option to allow a role's ability to update system catalog.

## Member of

In the grid, check **Granted** or **Admin Option** option against the role listed in **Role Name** to assign this role to be a member of selected role. Multiple roles can be granted.

## Member

In the grid, check **Granted** or **Admin Option** option against the role listed in **Member** to assign the selected role to be a member of this role. Multiple roles can be granted.

## Privileges

To edit specific object privileges for role, click  **Add Privilege** to open the window and follow the steps below:

1. Expand the node in the tree view until reaching to the target object.
2. Check the object to show the grid on the right panel.
3. In the grid, check **Granted** or **Grant Option** option against the privilege listed in **Privilege** to assign this role to have that privilege. Multiple privileges can be granted.

## SQLite Security

By default, a SQLite database does not require user authentication (no-authentication-required database). After you created a user, the database will be marked as requiring authentication (authentication- required database). Then, user need to provide username and password when connecting to the database file.

## SQLite User Designer

### User name

Set name for user.

### Password

Set **Password** and re-type it in the **Confirm Password** text box.


### Administrator


Check this option to give the admin privilege to the user.

## Privilege Manager

Besides setting privileges in each user, the **Privilege Manager** provides another view on privileges in connection and its database objects.

**Note:** Available only for MySQL, Oracle, PostgreSQL and MariaDB.


Click  **Privilege Manager** from the user object list toolbar and follow the steps below:


1. Expand the node in the tree view until reaching to the target object.
2. Choose the object and click  **Add Privilege** to open the window.
3. Check the user to show the grid on the right panel.
4. In the grid, check the relevant privilege against the privilege listed in **Privilege** to assign the selected user to have that object privilege. Multiple privileges can be granted.

# Useful Tools

Navicat provides variety of tools that improve user experience when using Navicat, which are **Object Information**, **Connection Colorings**, **Search Filter** and more.

## List/Detail/ER Diagram View

Navicat provides three types of views for objects in the main window. By default, Navicat uses the **List** view in the Object List pane. It only shows object names. You can select **View -> List** from the main menu or click  at the bottom right hand corner.

**Detail** view shows several properties of objects in columns. To change to Detail view, select **View -> Detail** from the main menu or click  at the bottom right hand corner.

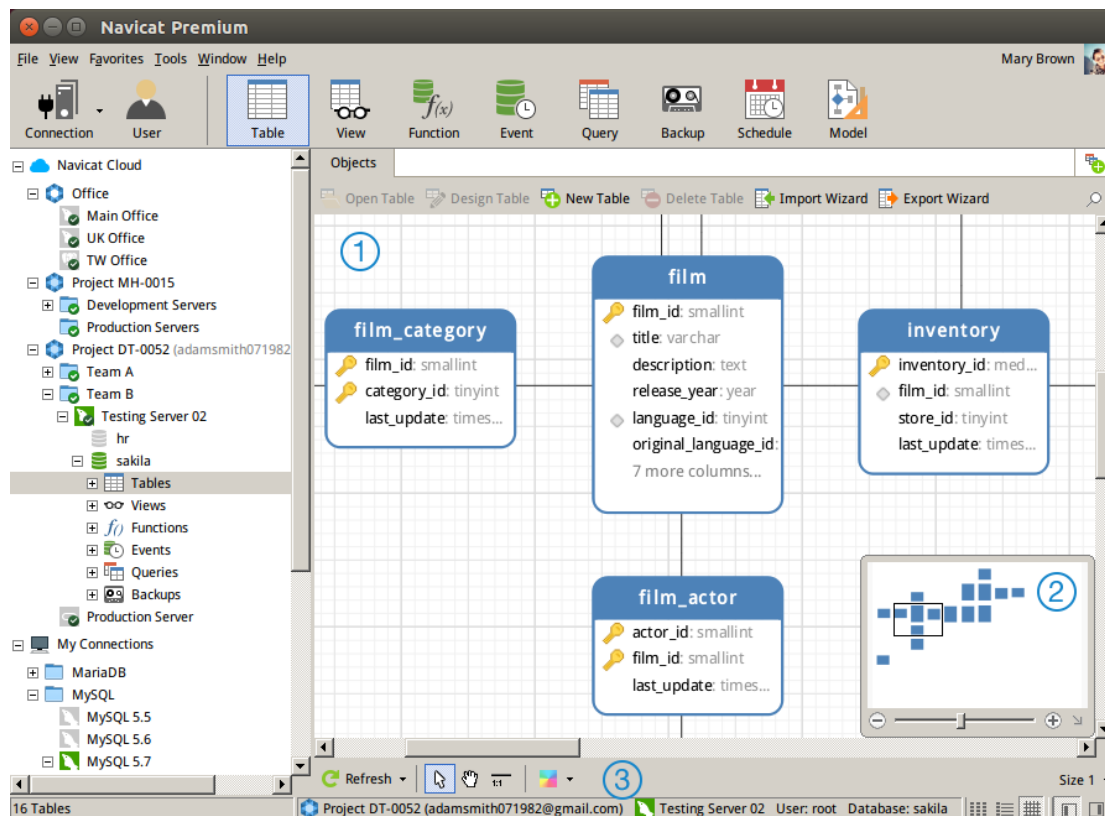
To change the display columns of properties, select **View -> Choose Columns** from the main menu and select display columns for different objects from the pop-up window.

Besides List and Detail views, Navicat enhances table viewing to a new **ER Diagram** view. In this ER Diagram view, you can view table fields and relationships between tables in a database/schema graphically. It also allows adding foreign key constraints to tables directly.

**Note:** Available only in Full Version. Only tables provide ER Diagram view. Other database objects only provide List view and Detail view.

Select **View -> ER Diagram** from the main menu or click  at the bottom right hand corner. An ER diagram will be created automatically if the selected database/schema contains tables.

**Hint:** ER Diagram files are stored under [Settings Location](#).



## ① Object List

Display table fields and relationships between tables in a database/schema.

**Note:** Double-click a table in ER Diagram view will open the Table Designer, while double-click a table in List view and Detail view will open the Table Viewer. The tabs and options in the designer depend on the diagram database type you have chosen. For the settings of different tabs, see [Server Objects](#).

To add a relation, click  from the bottom toolbar. Drag the source table field and drop to the target table field.

To edit a relation, right-click a relation and select **Design Foreign Key** from the pop-up menu.

To delete a relation, right-click a relation and select **Delete Foreign Key** from the pop-up menu.

To add a vertex on a relation, right-click a relation and select **Add Vertex** from the pop-up menu.

To delete a vertex or all vertices on a relation, right-click a relation and select **Delete Vertex** or **Delete All Vertices** from the pop-up menu.

## ② Overview

To zoom in or zoom out the selected area of the diagram, adjust the slider of the Overview pane. Same effect can be achieved with keyboard shortcuts:

Zoom In: [CTRL++] or [CTRL+Mousewheel up]

Zoom Out: [CTRL+-] or [CTRL+Mousewheel down]

## ③ Toolbar



## Refresh

Click to refresh the ER Diagram.

## Regenerate ER Diagram

Choose **Regenerate ER Diagram** from  **Refresh** button. It regenerates the ER Diagram with using auto layout feature.

## Move Diagram

Click to switch to hand mode. Press and hold the SPACE key, then move the diagram.

## New Relation

Click to create a relation between two table fields.

## Color

Set color to selected tables and relations.

## Paper Size

Select paper size from drop-down list, corresponding paper size will reflect in the Overview pane.

# Object Information

In the Object List pane, you can also view information of a selected object. Select **View -> Show Object Information** from the main menu or right-click the object and select **Object Information** from the pop-up menu.

**Note:** The tabs depend on the object type you have chosen.

## General

Show the object information.

## DDL

Show the DDL statement of the object.

## Using

Show the objects that the current object used.

## Used by

Show the current object used by whom.

## Objects

Show the objects in the tablespace.

## Preview

Show the sql statement in the query.


## Member of

Show the roles that the user or the role assigned to.

## Members

Show the members of the role.


# Server Monitor (Available only in Full Version)

Navicat provides **Server Monitor** to view properties of selected server(s). Select **Tools ->  Server Monitor** and select the preferred server type from the main menu.

**Note:** Available only for MySQL, Oracle, PostgreSQL and MariaDB.

## Process List

Display a list of processes from all servers selected.

To stop the selected process, click  **End Process** button.

## Auto refresh

If you want to take action on auto-refreshing the server in assigned seconds, choose **View -> Set Auto Refresh Time** and enter an auto refresh value. To disable auto refresh feature, choose **View -> Auto Refresh**.


**Note:** Effect will take once you assign the value.

The process list provides the following information depends on the database type you are chosen: MySQL, Oracle, PostgreSQL and MariaDB.

- Server name that is given while setting the connection.
- Process ID on the server.
- Serial number of the process.
- Current user who log in to the server.
- Host from which the user is connected.
- Database that the user is currently used.
- Last command that was issued by the user.
- Time, state and info of the process.

## Variables

Display the list of all server variables and their values.

**Hint:** To edit variable value in MySQL and Oracle servers, click  or press CTRL+ENTER to open editor for editing. The value in PostgreSQL server cannot be edited here. (Those variables can be set using the *SET* statement, by editing the *postgresql.conf* configuration file.)

## Status

Display the list of all server status and their values.

## Virtual Grouping (Available only in Full Version)

**Virtual Group** aims to provide a platform for logical grouping objects by categories, so that all objects are effectively preserved.

Virtual Grouping can be applied on Connection, Table, View, Function, Query, Backup, Schedule and Model.

Right-click in the Connection pane/the Object List pane and select **New Group** or **Manage Group** -> **New Group** from the pop-up menu to create a new group.

To move object(s) to a group, right-click the object(s) and select **Manage Group** -> **Move To** from the pop-up menu or drag and drop the object(s) into the group.

To move object(s) back to the top-level, right-click the object(s) and select **Manage Group** -> **Exclude From Group** from the pop-up menu or drag and drop the object(s) to target level in the Connection pane.

Select **View** -> **Flatten Connection/Flatten Object List** from the main menu to hide connection/object groups.

## Connection Colorings

Navicat provides highlighting connections by colors for identifying connections and their database objects. The highlighted color displays in the connection tree and menu bar in its database object window.

To highlight a connection, right-click the connection and select **Color** from the pop-up menu.

## Favorites (Available only in Full Version)

**Favorites** are links to database objects that you visit frequently. By adding a path to your favorites list, you can go to that database objects with a single click, instead of having to navigate the connection and database and/or schema in the connection tree.

To add a link to the favorites list, open a database object and choose **File** -> **Add To Favorites** or press SHIFT+CTRL+#. If database object is opened in tabbed window, you can also right-click the tab and select **Add To Favorites** from the pop-up menu. Enter **Favorite Name** and select **Favorite ID**.

To open a database object from the favorites list, select **Favorites** -> **favorite\_name** from the main menu or press CTRL+#.

Select **Favorites** -> **Clear Favorites** -> **favorite\_name** from the main menu to remove a link from the favorites list.

Select **Favorites** -> **Clear Favorites** -> **Clear All** from the main menu to remove all links from the favorites list.

**Note:** # represents 0, 1, 2, 3, 4, 5, 6, 7, 8 or 9.

## Find in Database/Schema (Available only in Full Version)

Navicat provides a **Find in Database/Schema** feature offers searching table and view records or object structures within a database and/or schema. Select **Tools** -> **Find in Database/Schema** from the main menu.

1. Select a target **Connection**, **Database** and/or **Schema**.
2. Enter the search string in **Find what**.
3. Choose to find Data or Structure in the **Look in** drop-down list.
4. Choose the **Search Mode**: Contains, Whole Word, Prefix or Regular Expression. Regular Expression is only available when finding Structure.
5. Check the **Case Insensitive** to disable case sensitive search if necessary.
6. When finding Structure, you can choose to search different objects: Tables, Views, Functions, Queries, Indexes, Triggers, Events and/or Materialized Views.
7. Click the **Find** button and then double-click an object in the **Find Results** list to view the record or the structure.

## Search Filter

Navicat provides search filters for searching your objects in the Connection pane, the Object List pane, the Model Designer window and other tree structures.

In the Connection pane or other tree structures, click on the pane or the tree to focus and enter a filter string directly. If connections have opened in the Connection pane, the filter will also apply to their database objects.

In the Object List pane, click  in the Navicat main window and enter a filter string in the Search box.

In the Model Designer window, simply enter a filter string in the Search box.

You can remove the filter by deleting the filter string.

Navicat Premium

File View Favorites Tools Window Help

Connection User Table View Function Event Query Backup Schedule Model

Navicat Cloud

- Project DT-0052 (adamsmith071982@gmail.com)
  - Team B
    - Testing Server 02
      - sakila
        - Tables
          - film
          - film\_actor
          - film\_category
          - film\_text
        - Views
        - Functions
        - Events
        - Queries
        - Backups

- My Connections
- MySQL
  - MySQL 5.7
    - sakila
      - Tables
        - film
        - film\_actor
        - film\_category
        - film\_text
      - Views
        - film\_list

Objects

Open Table Design Table New Table

film

Name	Auto Increment Value	Modified Date	Data Length	Table Type	Rows
film	1001		192 KB	InnoDB	808
film_actor			192 KB	InnoDB	4748
film_category			64 KB	InnoDB	829
film_text			176 KB	InnoDB	1039

film

4 item(s) found

Project DT-0052 (adamsmith071982@gmail.com) Testing Server 02 User: root Database

# Options

Navicat provides a complete user interface customization with various options for all tools. Select **Tools -> Options** from the main menu.

## General Options

### Allow Multiple Form Instances

With this option is on, you allow opening multiple instances of the same selected window.

### Allow Multiple Navicat Instances

Uncheck this item means that clicking on the Navicat shortcut will re-activate the running instance of Navicat and not launch a new copy.

### Click to refresh

Refresh the Object List pane whenever you click on the objects.

### Show function wizard

Display the function wizard ([MySQL/MariaDB](#), [Oracle](#) or [PostgreSQL](#)) when you create a new function/procedure.

### Ask to save new queries/profiles before closing

With this option is on, Navicat will prompt you to save new queries or profiles every time when you quit the relevant sub-window.

## Windows

### Open Windows

Open new windows to **main window** or **opened window** or **as a new window**.

## Code Insight (Available only in Full Version)

### Use Code Completion

When you type the . (dot) symbol between the object names, SQL Editor will offer you a pop-up list that showing some variants for the code completion, see [Code Completion](#).

### Delay

You can change the time the pop-up list takes to appear.

### Use Word Completion

When you type the first character of words, SQL Editor will offer you a pop-up list that showing some variants for the word completion.

### Delay

You can change the time the pop-up list takes to appear.

## Use Syntax Highlighting

Syntax highlight helps viewing codes clearly. Codes are highlighted in SQL Editor with different colors and fonts according to the categories they belong to. The syntax highlighting feature can be limited by setting the maximum file size (e.g. 10) in **Apply Syntax Highlighting for statement size below (MB)** to increase performance.

## Auto Save

### Use Auto Save

Save automatically after modifications in SQL Editor by defining the **Auto Save Interval (s)** (e.g. 30).

# Appearance Options

### Show toolbar caption

Show text on toolbar buttons in sub-windows. Otherwise, only buttons will be presented.

**Hint:** Restart Navicat to take effect.



### Show main toolbar caption

Show text on toolbar buttons in the main window. Otherwise, only buttons will be presented.

### Use big icons for main toolbar

Use large icons on toolbar buttons in the main window.

## Font

### GUI

Define the font and its size used by Navicat interface.

**Hint:** Restart Navicat to take effect.

### Grid Font

Define the font and its size used by grid in Table Viewer.

### Editor Font

Define the font and its size used by editors.

### ER Diagram Font

Define the font and its size used in ER diagram.

### Anti Alias

Check this option to improve the quality of the displaying fonts.

## Color

### Grid Colors

Define the background colors of the table grid.

### Use three colors

Display the table grid background by using three different colors respectively for viewing data clearly.

### Text Colors

This color settings allows you to format your SQL queries in SQL Editor with colored syntax highlighting for your SQL statements to improve readability.

Set font colors of the SQL Editor uses to mark out different text fragments: Common, Keywords, Comments, Strings and Numbers. Just simply click on the color boxes and choose your desired color from the **Color-Selection** dialog window.

## Main Window

### Show table hint

While you roll the mouse pointer over a table within the Object List pane, you could get a pop-up hint giving details about the table structure.

### Show objects in Connection pane

Display database objects using the tree structure in the connection tree. To expand a node, simply double-click the node.

**Hint:** Reopen the database/schema to take effect.

### Show system items (PostgreSQL)

Check this option to show all the system objects such as *information\_schema* and *pg\_catalog* schemas.

**Hint:** Reopen the database/schema to take effect.

### Show auto index (SQLite)

Check this option to show auto index generated for SQLite table in the Index's Object List pane.

## Editor

### Show Line Number

Display line numbers at the side of the editor for easily reference.

### Use Code Folding

Code folding allows codes to collapse as a block and only the first line displayed in the editor, see [Code Folding](#).

### Use Brace Highlighting

Highlight the pair of braces when your cursor moves to either one brace for easily reference, see [Brace Highlight](#).



## Tab Width

Enter the number of characters that a tab occupies, e.g. 5.

## Data & Grid

### Show TEXT Blob fields in data grids

If this option is on, data which set as TEXT field type is visible in table grid. Otherwise, [\(WIDEMEMO\)](#) will be shown.

### Limit Records

Check this option if you want to limit the number of records showed on each page in table grid/foreign key data selection globally. Otherwise, all records will be displayed in one single page.

#### ☐ records per page

Set the **records per page** value (e.g. 1000) in the edit field. The number representing the number of records showed per page in table grid.

**Note:** To adjust the settings for particular table, see [Table Viewer](#).

#### ☐ records per page in foreignkey editor

Set the **records per page in foreignkey editor** value (e.g. 100) in the edit field. The number representing the number of records showed per page in [Foreign Key Data Selection](#).

### Synchronize current record

When updating/inserting a row in table grid (if primary key exists), it will reload this record from server.

For example: a table with 3 columns - id, name and timestamp. If you update the name column, the timestamp will update immediately in the grid.

### Row height

Define the height of the row (e.g. 17) used in editor.




**Note:** To adjust the settings for particular table, see [Formatting Table Grid](#).

### Column Width

Define the width of the column (e.g. 150) used in editor.

**Note:** To adjust the settings for particular table, see [Formatting Table Grid](#).

### Auto Begin Transaction

Check this option if you require auto commit of changing records in table grid. Otherwise, click  **Commit** or  **Rollback** buttons from the  **Begin Transaction** button to commit or rollback the changes. See [Table Viewer](#).

## Display Formats

Data of types integer, float, date, time and datetime can be formatted when displayed on data grids. Type the format here to change the format. If the formats are left blank, default format will be used. For date, time and datetime fields, default formats will be the system datetime formats.

Display formats

Integer:

Float:

Date:

Time:

DateTime:

Example:
11/18/2015 11:44:55 AM

Output:
18/11/15 11:44:55

Formats are defined by constructing a string using these format specifiers:

## Numeric fields

Specifier	Represent
0	Digit placeholder. If the value being formatted has a digit in the position where the "0" appears in the format string, then that digit is copied to the output string. Otherwise, a "0" is stored in that position in the output string. (e.g. with 0000 placed in the Integer field, all the integer output from the table will have 0012 in format)
#	Digit placeholder. If the value being formatted has a digit in the position where the "#" appears in the format string, then that digit is copied to the output string. Otherwise, nothing is stored in that position in the output string. (e.g. with ## ## placed in the Integer field, all the integers output from the table will have 12 34 in format)
.	Decimal point. The first "." character in the format string determines the location of the decimal separator in the formatted value; any additional "." characters are ignored. The actual character used as a the decimal separator in the output string is determined by the DecimalSeparator global variable.
,	Thousand separator. If the format string contains one or more "," characters, the output will have thousand separators inserted between each group of three digits to the left of the decimal point. The placement and number of "," characters in the format string does not affect the output, except to indicate that thousand separators are wanted. The actual character used as a the thousand separator in the output is determined by the ThousandSeparator global variable.
E+	Scientific notation. If any of the strings "E+", "E-", "e+", or "e-" are contained in the format string, the number is formatted using scientific notation. A group of up to four "0" characters can immediately follow the "E+", "E-", "e+", or "e-" to determine the minimum number of digits in the exponent. The "E+" and "e+" formats cause a plus sign to be output for positive exponents and a minus sign to be output for negative exponents. The "E-" and "e-" formats output a sign character only for negative exponents.
'xx'/"xx"	Characters enclosed in single or double quotes are output as-is, and do not affect

	formatting.
;	Separates sections for positive, negative, and zero numbers in the format string.

The locations of the leftmost "0" before the decimal point in the format string and the rightmost "0" after the decimal point in the format string determine the range of digits that are always present in the output string.

The number being formatted is always rounded to as many decimal places as there are digit placeholders ("0" or "#") to the right of the decimal point. If the format contains no decimal point, the value being formatted is rounded to the nearest whole number.

If the number being formatted has more digits to the left of the decimal separator than there are digit placeholders to the left of the "." character in the format string, the extra digits are output before the first digit placeholder.

The following table shows the effect of various format strings:

Display Format	Value	Result	Comment
###	12.2	12.2	Note extra digit to left of decimal still appears.
#.00	2.5	2.50	Note extra zero: field will always show two decimal places.
00.##	.006	00.01	Note extra 0s to right of decimal point and rounding to two decimal places.

To allow different formats for positive, negative, and zero values, the format string can contain between one and three sections separated by semicolons.

**One section:** The format string applies to all values.

**Two sections:** The first section applies to positive values and zeros, and the second section applies to negative values.

**Three sections:** The first section applies to positive values, the second applies to negative values, and the third applies to zeros.

If the section for negative values or the section for zero values is empty, that is, if there is nothing between the semicolons that delimit the section, the section for positive values is used instead.

If the section for positive values is empty, or if the entire format string is empty, the value is formatted using general floating-point formatting with 15 significant digits. General floating-point formatting is also used if the value has more than 18 digits to the left of the decimal point and the format string does not specify scientific notation.

## Date Time fields

Specifier	Represent
c	The date using the format given by the ShortDateFormat global variable, followed by the time using the format given by the LongTimeFormat global variable. The time is not displayed if the fractional part of the DateTime value is zero.
d	The day as a number without a leading zero (1-31).
dd	The day as a number with a leading zero (01-31).
ddd	The day as an abbreviation (Sun-Sat) using the strings given by the ShortDayNames global variable.

dddd	The day as a full name (Sunday-Saturday) using the strings given by the LongDayNames global variable.
dddddd	The date using the format given by the ShortDateFormat global variable.
ddddddd	The date using the format given by the LongDateFormat global variable.
m	The month as a number without a leading zero (1-12). If the m specifier immediately follows an h or hh specifier, the minute rather than the month is displayed.
mm	The month as a number with a leading zero (01-12). If the mm specifier immediately follows an h or hh specifier, the minute rather than the month is displayed.
mmm	The month as an abbreviation (Jan-Dec) using the strings given by the ShortMonthNames global variable.
mmmm	The month as a full name (January-December) using the strings given by the LongMonthNames global variable.
yy	The year as a two-digit number (00-99).
yyyy	The year as a four-digit number (0000-9999).
h	The hour without a leading zero (0-23).
hh	The hour with a leading zero (00-23).
n	The minute without a leading zero (0-59).
nn	The minute with a leading zero (00-59).
s	The second without a leading zero (0-59).
ss	The second with a leading zero (00-59).
t	The time using the format given by the ShortTimeFormat global variable.
tt	The time using the format given by the LongTimeFormat global variable.
am/pm	The time using the 12-hour clock for the preceding h or hh specifier, followed by "am" for any hour before noon, or "pm" for any hour after noon. The am/pm specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
a/p	The time using the 12-hour clock for the preceding h or hh specifier, followed by "a" for any hour before noon, or "p" for any hour after noon. The a/p specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
ampm	The time using the 12-hour clock for the preceding h or hh specifier, followed by the contents of the TimeAMString global variable for any hour before noon, or the contents of the TimePMString global variable for any hour after noon.
/	The date separator character given by the DateSeparator global variable.
:	The time separator character given by the TimeSeparator global variable.
'xx'/'xx'	Characters enclosed in single or double quotes are displayed as-is, with no formatting changes.

Format specifiers may be written in uppercase or lowercase letters; both produce the same result.

## Model Options (Available only in Full Version)

### Highlight Objects

With this option is on, when a mouse cursor hovers over an object, Navicat will highlight its border with blue color.

**Hint:** Restart Navicat to take effect.

## Highlight With Relation

With this option is on, when a mouse cursor hovers over a table or a view, Navicat will highlight it's foreign keys or view relations with blue or green color indicating relationships with other objects.

**Hint:** Restart Navicat to take effect.

## Guess Field Type

With this option is on, Navicat will [predict field types](#) when you design fields in tables.

**Hint:** Restart Navicat to take effect.

# Miscellaneous Options

## Files Location

You can change the folder for different types of files. By default, most of the files are stored in [Settings Location](#). However, some files are located in the [profiles](#) directory and all the [log files](#) are stored in the **logs** directory.

## Activation & Updater

### Check for updates on startup

Check this option to allow Navicat checks for new version when it starts.

## Proxy

### Use Proxy

Check this option to use proxy for the activation process and enter **Host**, **Port**, **User Name** and **Password**.

## OCI (Available only for Oracle)

### OCI library (oci.dll)

Choose the **Oracle Client** / **Oracle Instant Client** folder path that includes the OCI library (oci.dll) for [Basic](#) connection. It has already included in Navicat installation folder. You can locate it directly.

**Oracle Instant Client** is the simplest way to deploy a full Oracle Client application built with OCI, OCCI, JDBC-OCI, or ODBC drivers. It provides the necessary Oracle Client libraries in a small set of files. You can also download **Oracle Client** / **Oracle Instant Client** through -

### Oracle Client

<http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html>

### Oracle Instant Client

<http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html>

Download the appropriate Instant Client packages for your platform and the CPU. All installations REQUIRE the Basic or Basic Lite package. Unzip the packages and set the path points to it.

## Oracle Client Installation Guide

[http://download.oracle.com/docs/cd/B28359\\_01/install.111/b32302/toc.htm](http://download.oracle.com/docs/cd/B28359_01/install.111/b32302/toc.htm)

**Hint:** Restart Navicat to take effect.

# Commands (Available only in Full Version)

Navicat Objects	Server Type	Command Lines
Backup	MySQL, PostgreSQL, SQLite and MariaDB	start_navicat -backup [ProfileName] -u NavicatID -p ProjectName -t ConnectionType -c ConnectionName -d DatabaseName -s SchemaName
Import	All	start_navicat -import ProfileName -u NavicatID -p ProjectName -t ConnectionType -c ConnectionName -d DatabaseName -s SchemaName
Export Table	All	start_navicat -export ProfileName -u NavicatID -p ProjectName -t ConnectionType -c ConnectionName -d DatabaseName -s SchemaName
Export View Result	All	start_navicat -exportview ProfileName -u NavicatID -p ProjectName -t ConnectionType -c ConnectionName -d DatabaseName -s SchemaName
Export Materialized View Result	Oracle and PostgreSQL	start_navicat -exportmview ProfileName -u NavicatID -p ProjectName -t ConnectionType -c ConnectionName -d DatabaseName -s SchemaName
Export Query Result	All	start_navicat -exportquery ProfileName -u NavicatID -p ProjectName -t ConnectionType -c ConnectionName -d DatabaseName -s SchemaName
Query Execution	All	start_navicat -query QueryName -u NavicatID -p ProjectName -t ConnectionType -c ConnectionName -d DatabaseName -s SchemaName
Data Transfer	All	start_navicat -datatransfer ProfileName -t ProfileType
Data Synchronization	All	start_navicat -datasync ProfileName -t ConnectionType
Batch Jobs	All	start_navicat -batchjob BatchJobName

## Note:

NavicatID - if the connection stores in Navicat Cloud, Navicat ID is required, e.g. user@example.com

ProjectName - if the connection stores in Navicat Cloud, project name is required

ConnectionType- type of the connection: MySQL, Oracle, PostgreSQL, SQLite or MariaDB

ProfileType - type of the data transfer profile: MySQL, Oracle, PostgreSQL, SQLite, MariaDB or Premium

Example:

```
start_navicat -exportquery MyQueryExport1 -u test@navicat.com -p Project1 -t MySQL -c "MySQL 5.6" -d sakila
```





# Hot Keys

## Navicat Main Window

Keys	Action
CTRL+# (# represents 0 to 9)	Open Object Window from Favorites List
CTRL+H	History Log
CTRL+Q	New Query
F12	Show Only Active Objects

## Common

Keys	Action
CTRL+N	New Object
SHIFT+CTRL+# (# represents 0 to 9)	Add to Favorites
F8	Navicat Main Window
CTRL+TAB or SHIFT+CTRL+TAB	Next Window
F1	Help
CTRL+F1	Online Documentation

## Table Designer

Keys	Action
CTRL+O	Open Table
CTRL+F	Find Field
F3	Find Next Field
SHIFT+F3	Find Previous Field

## Table Viewer/View Viewer

Keys	Action
CTRL+D	Design Table/Design View
CTRL+Q	Query Table/Query View
CTRL+F	Find Text
F3	Find Next Text
CTRL+G	Go to Row
CTRL+LEFT ARROW	First Data Column of Current Record
CTRL+RIGHT ARROW	Last Data Column of Current Record
CTRL+HOME	First Data Row of Current Column
CTRL+END	Last Data Row of Current Column
CTRL+PAGE UP or CTRL+UP ARROW	First Data Row of Current Window
CTRL+PAGE DOWN or CTRL+DOWN ARROW	Last Data Row of Current Window

CTRL+R	Apply Filter in Filter Wizard
SHIFT+ARROW	Select Cells
CTRL+ENTER	Edit Data with Opening Editor
INSERT or CTRL+N	Insert Record
CTRL+DELETE	Delete Record
CTRL+S	Apply Record Changes
ESC	Cancel Record Changes
CTRL+T	Stop Loading Data

## View/Query

Keys	Action
CTRL+O	Load View/Load Query
CTRL+/ SHIFT+CTRL+/ CTRL+E	Comment Line Uncomment Line View Definition/Query Editor
CTRL+R	Run
SHIFT+CTRL+R	Run Selected
F7	Run a Statement from Here
CTRL+T	Stop

## SQL Editor

Keys	Action
CTRL+F	Find Text
F3	Find Next Text
CTRL+= or CTRL+Mousewheel Up	Zoom In
CTRL+- or CTRL+Mousewheel Down	Zoom Out
CTRL+0	Reset Zoom

## Debugger

Keys	Action
F9	Run
F8	Step Over
F7	Step In
SHIFT+F7	Step Out

## Model

Keys	Action
CTRL+D	New Diagram in Model
CTRL+P	Print

ESC	Select
H	Move Diagram
T	New Table
V	New View
L	New Layer
A	New Label
N	New Note
I	New Image
R	New Foreign Key
CTRL+B	Bold Selected Table, View, Foreign Key or Shape
CTRL+= or CTRL+Mousewheel Up	Zoom In
CTRL+- or CTRL+Mousewheel Down	Zoom Out
CTRL+0	Reset Zoom

# Log Files

Navicat provides number of log files to keep track on the actions have been performed in Navicat and they are located in the **logs** directory, e.g. Y:\.navicat\Navicat\MySQL\logs\. You are allowed to change the log files location under [Options](#).

## HttpDump.log

Store information which response from your HTTP Server.

## LogHistory.txt

Store all SQL statements of all the operations executed over databases and database objects in Navicat. Select **Tools** -> **History Log** from the main menu or press CTRL+H to open the LogHistory.txt file in the History Log Viewer.

**Note:** This log will be overwritten while Navicat is being restarted.

## LogImport.txt

Record detailed information on every error (indicating success or failure) that occurred during the import process.

**Note:** This log will be overwritten on each import.

## LogExport.txt

Record detailed information on every error (indicating success or failure) that occurred during the export process.

**Note:** This log will be overwritten on each export.

## LogSynchronize.txt

Record detailed information on every error (indicating success or failure) that occurred during the data synchronization process.

**Note:** This log will be overwritten on each synchronization.

## LogCmd.txt

Store information for Navicat command line process and all operations while running schedule.