

Table of Contents

MYSQL DATABASE OBJECT MANAGEMENT	2
MYSQL TABLES	3
<i>MySQL Table Designer</i>	7
MySQL Table Fields	8
Setting MySQL Table Field Properties	10
Setting Other MySQL Table Field Properties	12
MySQL Table Indexes	14
Setting MySQL Table Index Properties	15
MySQL Table Foreign Keys	17
Setting MySQL Table Foreign Key Properties	19
MySQL Table Triggers	20
Setting MySQL Table Trigger Properties	21
MySQL Table Options	22
Setting MySQL Table Partition Options	25
MYSQL VIEWS	27
<i>MySQL View Designer</i>	30
Working with MySQL View Builder (Available only in Full Version)	31
Editing MySQL View SQL Definition	32
Setting Advanced MySQL View Properties	33
MySQL View Preview	35
MySQL View Explain	36
<i>MySQL View Viewer</i>	37
MYSQL FUNCTIONS/PROCEDURES	38
<i>MySQL Function Wizard</i>	41
Setting MySQL Routine Type	42
Setting Parameters for MySQL Procedure/Function	43
Setting Return Type for MySQL Function	44
<i>MySQL Function/Procedure Designer</i>	45
Editing MySQL Function/Procedure Definition	46
Setting Advanced MySQL Function/Procedure Properties	47
Viewing MySQL Function/Procedure Result	48
MYSQL EVENTS	49
<i>MySQL Event Designer</i>	51
Editing MySQL Event Definition	52
Setting Advanced MySQL Event Properties	53


MySQL Database Object Management

The following list contains the most common MySQL database objects supported by Navicat.

- [Tables](#)
- [Views](#)
- [Functions/Procedures](#)
- [Events](#)



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.

Just simply click  to open an object pane for **Table**. A right-click displays the popup menu or using the object pane toolbar, allowing you to create new, edit, open and delete the selected table.

Create Table

To create a new table

- Select anywhere on the object pane.
- Click the  **New Table** from the object pane toolbar.
or
- Right-click and select  **New Table** from the popup menu.
- Edit table properties and fields on the appropriate tabs of the Table Designer.




Hint: To create new table you can also right-click the Tables node of the navigation pane and select the  **New Table** from the popup menu.

To create a new table with the same properties as one of the existing tables has (using popup menu)

Apply to: current database {same connection}



- Select the table(s) for copying in the navigation pane/object pane.
- Right-click and select the **Duplicate Table** from the popup menu.
- The newly created table(s) will be named as "tablename_**copy**".

To create a new table with modification as one of the existing tables

- Select the table for modifying in the navigation pane/object pane.
- Right-click and select the  **Design Table** from the popup menu.
or
- Click the  **Design Table** from the object pane toolbar.
- Modify table properties and fields on the appropriate tabs of the Table Designer.
- Click  **Save As**.

Edit Table

To edit the existing table (manage its fields, indexes, foreign keys and triggers etc)



- Select the table for editing in the navigation pane/object pane.
- Right-click and select the  **Design Table** from the popup menu.
or
- Click the  **Design Table** from the object pane toolbar.
- Edit table properties and fields on the appropriate tabs of the Table Designer.


To change the name of the table

- Select the table for editing in the navigation pane/object pane.
- Right-click and select the **Rename** from the popup menu.


Open Table (manage table data)

To open a table

- Select the table for opening in the navigation pane/object pane.
- Right-click and select the  **Open Table** from the popup menu or simply double-click the table.
or
- Click the  **Open Table** from the object pane toolbar.

Note: This option is only applied if you do wish Navicat loads all your images while opening the table. To open the graphical table with faster performance, use  **Open Table (Quick)** below.

To open a table with graphical fields

- Select the table for opening in the navigation pane/object pane.
- Right-click and select the  **Open Table (Quick)** from the popup menu.

Note: Faster performance for opening the graphical table, as BLOB fields (images) will not be loaded until you click on the cell.

Empty Table

To empty a table

- Select the table in the navigation pane/object pane.
- Right-click the selected table and choose **Empty Table** from the popup menu.

Note: 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** below.



Truncate Table

To truncate a table

- Select the table in the navigation pane/object pane.
- Right-click the selected table and choose **Truncate Table** from the popup menu.

Delete Table

To delete a table

- Select the table for deleting in the navigation pane/object pane.
- Right-click and select the  **Delete Table** from the popup menu.
or
- Click the  **Delete Table** from the object pane toolbar.
- Confirm deleting in the dialog window.

Achieve Table Information

To achieve a table information

- Select the table in the navigation pane/object pane.
- Right-click the selected table and choose **Object Information** from the popup menu.
or
- Choose View -> Object Information in the main menu.

MySQL Table Designer

Table Designer is the basic Navicat tool for working with tables. It allows you to create, edit and drop table's fields, indexes, foreign keys, and much more.



- [Managing Table Fields](#)
- [Managing Table Indexes](#)
- [Managing Table Foreign Keys](#)
- [Managing Table Triggers](#)
- [Managing Table Options](#)
- Managing Table Comment
- Table SQL Preview

MySQL Table Fields

Table fields are managed on the **Fields** tab of the Table Designer. Just simply click a field for editing. A right-click displays the popup menu or using the field toolbar, allowing you to create new, insert, move and drop the selected field.

Add Field

To add a field to the table



- Open the table in the Table Designer.
- Open the **Fields** tab.
- Right-click and select the  **Add Field** from the popup menu or click the  **Add Field** from the toolbar.
- Edit field properties.

To add a new field with modification as one of the existing fields

- Open the table in the Table Designer.
- Open the **Fields** tab.
- Select field.
- Right-click and select the **Duplicate Field** from the popup menu.
- Edit field properties.

Insert Field

To insert a field above an existing field

- Open the table in the Table Designer.
- Open the **Fields** tab.
- Select field.
- Right-click and select the  **Insert Field** from the popup menu or click the  **Insert Field** from the toolbar.
- Define field properties in the empty row.





Note: Support from MySQL 3.22 or later.

Edit Field

To edit the table field

- Open the table in the Table Designer.
- Open the **Fields** tab.
- Simply click on the field to edit.



To change the order of the table fields

- Open the table in the Table Designer.
- Open the **Fields** tab.
- Right-click on the field to move and select the  **Move Up**/  **Move Down** from the popup menu or click the  **Move Up**/  **Move Down** from the toolbar.


Note: Support from MySQL 4.0.1 or later.

Delete Field

To delete the table field

- Open the table in the Table Designer.
- Open the **Fields** tab.
- Right-click on the field to delete and select the  **Delete Field** from the popup menu or click the  **Delete Field** from the toolbar.
- Confirm deleting in the dialog window.

Setting MySQL Table Field Properties

Name	Type	Length	Decimals	Allow Null	
▶ OrderNo	double	0	0	<input type="checkbox"/>	 1
CustNo	double	0	0	<input checked="" type="checkbox"/>	
SaleDate	datetime	0	0	<input checked="" type="checkbox"/>	
ShipDate	datetime	0	0	<input checked="" type="checkbox"/>	
EmpNo	bigint	20	0	<input checked="" type="checkbox"/>	
ShipToContact	varchar	20	0	<input checked="" type="checkbox"/>	

Name

The Name is a descriptive identifier for a field that can be up to 64 characters (letters or numbers) including spaces. The names should be descriptive enough that anyone can easily identify them when viewing or editing records. For example, LastName, FirstName, StreetAddress, or HomePhone.

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.

Type

After you name a field, you choose a data type for the data to be contained in the field. When you choose a field's data type, you are deciding:

- What kind of values to allow in the field. You cannot store text in field with the **Numeric** data type.
- How much storage space MySQL is to set aside for the data in that field.
- What types of operations can be performed on the values in that field.

The **Type** dropdown list defines the type of the field data. See [MySQL Data Types](#) for details.

Length and Decimals

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.

Allow Null

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.

Setting Other MySQL Table Field Properties

To set the default value for the field use the **Default** edit box.

Note: TEXT(tinytext, text, mediumtext and longtext) and BLOB(tinyblob, blob, mediumblob and longblob) data type cannot have **DEFAULT** values.

To set any optional text describing the current field use the **Comment** edit box.

Note: Apply to all data type.

To set other field properties for Text/Memo and BLOB (Binary Large Object) (not apply to binary/varbinary type)

Character set (non-binary strings only)

A **character set** is a set of symbols and encodings. The **Character set** drop-down list defines the type of the character set for field.

Collation (non-binary strings only)

A **collation** is a set of rules for comparing characters in a character set. The **Collation** drop-down list defines the type of the collation for field.

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. KEY LENGTH (1 - 255).

Binary (char and varchar only)

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.

To set other field properties for Number/Currency and Floating Point (not apply to bit type)

Auto Increment (Number/Currency only)

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.

To set other field properties for Date/Time

On Update Current_Timestamp (timestamp only)

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

To set other field properties for Set/Enumerate

Values

Use **Values** edit box to define the members of SET/ENUM. A SET field can have a maximum of 64 members. An ENUM field can have a maximum of 65,535 distinct values.

MySQL 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).

Table indexes are managed on the **Indexes** tab of the Table Designer. Just simply click/double-click an index field for editing. A right-click displays the popup menu or using the index toolbar, allowing you to create new, edit and delete the selected index field.

Add Index

To add a table index

- Open the table in the Table Designer.
- Open the **Indexes** tab.
- Right-click and select the  **Add Index** from the popup menu or click the  **Add Index** from the toolbar.
- Edit index properties.



Edit Index

To edit a table index

- Open the table in the Table Designer.
- Open the **Indexes** tab.
- Just simply click/double-click on the index to edit.

Delete Index


To delete a table index

- Open the table in the Table Designer.
- Open the **Indexes** tab.
- Right-click on the index to delete and select the  **Delete Index** from the popup menu or click the  **Delete Index** from the toolbar.
- Confirm deleting in the dialog window.

Setting MySQL Table Index Properties

Name	Fields	Index Type	Index method
▶ CustNo	CustNo	...	Normal BTREE

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 (1 - 255).

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



The **Index Type** dropdown list defines 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.

MySQL 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.

Foreign Keys are managed on the **Foreign Keys** tab of the Table Designer. Just simply click/double-click a foreign key field for editing. A right-click displays the popup menu or using the foreign key toolbar, allowing you to create new, edit and delete the selected foreign key field.

Note: Foreign Key support from MySQL 3.23.44 or later.

Add Foreign Key

To add a foreign key

- Open the table in the Table Designer.
- Open the **Foreign Keys** tab.
- Right-click and select the  **Add Foreign Key** from the popup menu or click the  **Add Foreign Key** from the toolbar.
- Edit foreign key properties.

Note: Both tables must be *InnoDB* type (or *solidDB* type if you have [solidDB for MySQL](#)). 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.

Edit Foreign Key



To edit a foreign key

- Open the table in the Table Designer.
- Open the **Foreign Keys** tab.
- Just simply click/double-click on the foreign key to edit.

Note: Support from MySQL 4.0.13 or later.

Delete Foreign Key

To delete a foreign key

- Open the table in the Table Designer.
- Open the **Foreign Keys** tab.
- Right-click on the foreign key to delete and select the  **Delete Foreign Key** from the popup menu or click the  **Delete Foreign Key** from the toolbar.
- Confirm deleting in the dialog window.


Note: Support from MySQL 4.0.13 or later.

Setting MySQL Table Foreign Key Properties

Name	Fields	Reference Database	Reference Table	Reference Fields	On Delete	On Update
▶ cust_no_fk	CustNo	... report_sample	customer	CustNo	RESTRICT	RESTRICT

Use the **Name** edit box to enter a name for the new key and then select a table field to include in the key from the **Fields** group.

Use the **Reference Database** and **Reference Table** dropdown lists to select a foreign database and table respectively.

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

The **On Delete** and **On Update** dropdown 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.

MySQL 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.

Triggers are managed on the **Triggers** tab of the Table Designer. Just simply click a trigger field for editing. A right-click displays the popup menu or using the trigger toolbar, allowing you to create new, edit and delete the selected trigger field.

Note: Trigger is supported from MySQL 5.0.2 or later.

Add Trigger

To add a trigger

- Open the table in the Table Designer.
- Open the **Triggers** tab.
- Right-click and select the  **Add Trigger** from the popup menu or click the  **Add Trigger** from the toolbar.
- Edit trigger properties.



Edit Trigger

To edit a trigger

- Open the table in the Table Designer.
- Open the **Triggers** tab.
- Just simply click on the trigger to edit.

Delete Trigger

To delete a trigger

- Open the table in the Table Designer.
- Open the **Triggers** tab.
- Right-click on the trigger to delete and select the  **Delete Trigger** from the popup menu or click the  **Delete Trigger** from the toolbar.
- Confirm deleting in the dialog window.

Setting MySQL Table Trigger Properties

Name	Fires	Insert	Update	Delete
* tri1	After	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

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

Use the **Fires** dropdown 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 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

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

Note: Support *MyISAM* only.

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

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.

Note: Takes effect only with *MyISAM* tables.

Delay Key Write

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

Note: Support *MyISAM* only.

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.

Partition

Set the Partition Options.

Note: Support from MySQL 5.1 or later.

MRG_MYISAM table type

Union

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

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.

FEDERATED table type

Connection

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'
```

ndbcluster table type

Tablespace

To specify the tablespace for the storage.

Note: Support from MySQL 5.1.6 or later.

Storage

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.

Setting MySQL Table Partition Options

Partition By

Select the function that is used to determine the partition: **HASH**, **KEY**, **LINEAR HASH**, **LINEAR KEY**, **RANGE** and **LIST**.

Partitions

Set the partition number.

Subpartition By

Select the function that is used to determine the subpartition: **Hash** and **Key**.

Subpartitions

Set the subpartition number.

Partition Definition

Use or **Partition** to add or delete the partition. Use or **Subpartition** 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 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.

Just simply click  to open an object pane for **View**. A right-click displays the popup menu or using the object pane toolbar, allowing you to create new, edit, open and delete the selected view.




Create View

To create a new view




- Select anywhere on the object pane.
- Click the  **New View** from the object pane toolbar.
or
- Right-click and select  **New View** from the popup menu.
- Edit view properties on the appropriate tabs of the View Designer.

Hint: To create new view you can also right-click the Views node of the navigation pane and select the  **New View** from the popup menu.

To create a new view with modification as one of the existing views



- Select the view for modifying in the navigation pane/object pane.
- Right-click and select the  **Design View** from the popup menu.
or
- Click the  **Design View** from the object pane toolbar.
- Modify view properties on the appropriate tabs of the View Designer.
- Click  **Save As**.

To create a new view with loading from a SQL file

- Select anywhere on the object pane.
- Click the  **New View** from the object pane toolbar.
- or
- Right-click and select  **New View** from the popup menu.
- Click  **Load**.

Edit View

To edit the existing view (manage its SQL definition etc)



- Select the view for editing in the navigation pane/object pane.
- Right-click and select the  **Design View** from the popup menu.
- or
- Click the  **Design View** from the object pane toolbar.
- Edit view properties on the appropriate tabs of the View Designer.

To change the name of the view

- Select the view for editing in the navigation pane/object pane.
- Right-click and select the **Rename** from the popup menu.



Open View

To open a view (manage view data)

- Select the view for opening in the navigation pane/object pane.
- Right-click and select the  **Open View** from the popup menu or simply double-click the view.
- or
- Click the  **Open View** from the object pane toolbar.

Delete View

To delete a view

- Select the view for deleting in the navigation pane/object pane.
- Right-click and select the  **Delete View** from the popup menu.
or
- Click the  **Delete View** from the object pane toolbar.
- Confirm deleting in the dialog window.

Achieve View Information

To achieve a view information

- Select the view in the navigation pane/object pane.
- Right-click the selected view and choose **Object Information** from the popup menu.
or
- Choose View -> Object Information in the main menu.

MySQL View Designer

View Designer is the basic Navicat tool for working with views. It allows you to create new view and edit the existing view definition (view name and the SELECT statement it implements).

- [Working with View Builder](#)
- [Editing View SQL Definition](#)
- [Setting Advanced View Properties](#)
- View SQL Preview
- [View Preview](#)
- [View Explain](#)

Working with MySQL 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.

Editing MySQL View SQL Definition

The **Definition** tab allows you to edit the view definition as SQL statement (SELECT statement it implements).

Example:

```
SELECT
  clients.RecordID
FROM
  clients
```

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

Setting Advanced MySQL View Properties

Algorithm

Algorithm is optional and it is a MySQL extension to standard SQL. Algorithm takes three values: **Undefined**, **Merge** or **Temptable**. The default algorithm is Undefined if no Algorithm clause is present. The algorithm affects how MySQL processes the view.

Undefined

For **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

For **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

For **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


The **Check option** can be given for an updatable view to prevent inserts or updates to rows except those for which the *WHERE* clause in the *select_statement* is true. The **Local** and **Cascaded** keywords determine the scope of check testing when the view is defined in terms of another view. **Local** restricts the Check option only to the view being defined. **Cascaded** causes the checks for underlying views to be evaluated as well. When neither keyword is given, the default is **Cascaded**.

MySQL View Preview

To preview the result of the view, click  **Preview** on the toolbar. If the query statement is correct, the **Result** and **Message** tabs will be opened.

The **Result** tab displays the data of the view as a grid and the **Message** tab displays the message log.

MySQL View Explain

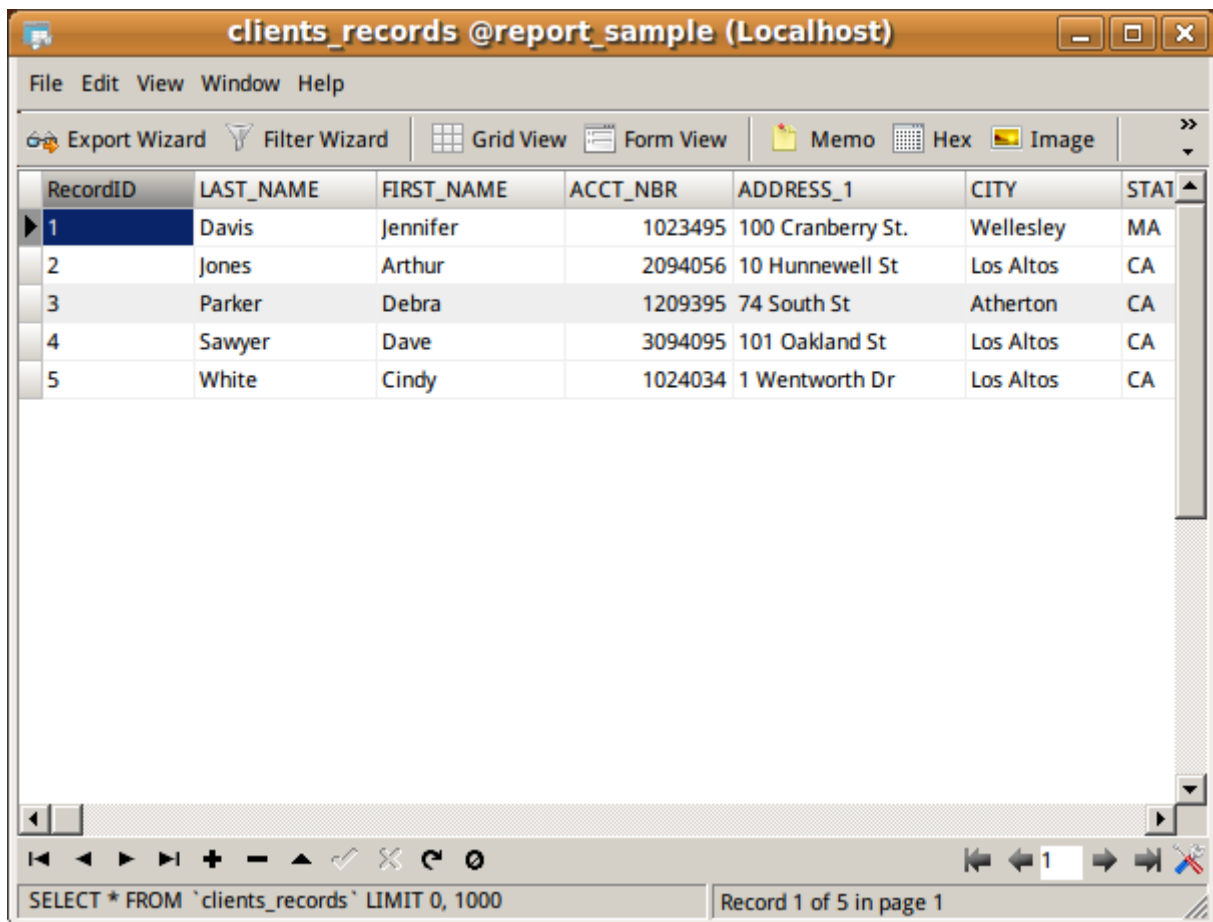
To show the Query Plan of the view, click  **Explain** on the toolbar. If the query statement is correct, the **Explain** tab will show the query plan.

MySQL View Viewer

View Viewer displays the view data as a grid. Data can be displayed in three modes:  **Grid View**,  **Form View** and **Text/Blob View**. See Data View for details.


The toolbars of View Viewer provides the following functions for managing data:

- **Export Data**
Export data to TXT, DBF, HTML, SQL, RTF and more.
- **Filter Data**
Allow you to filter records by creating and applying filter criteria for the data grid.
- **Edit TEXT/BLOB**
Allow you to view and edit the content of TEXT and BLOB fields.





MySQL 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.

Just simply click  to open an object pane for **Function**. A right-click displays the popup menu or using the object pane toolbar, allowing you to create new, edit and delete the selected function/procedure.




Create Function/Procedure

To create a new function/procedure

- Select anywhere on the object pane.
- Click the  **New Function** from the object pane toolbar.
or
- Right-click and select  **New Function** from the popup menu.
- Edit function/procedure properties on the appropriate tabs of the Function/Procedure Designer.



Hint: To create new function/procedure you can also right-click the Function node of the navigation pane and select the  **New Function** from the popup menu.

To create a new function/procedure with modification as one of the existing function/procedure

- Select the function/procedure for modifying in the navigation pane/object pane.
- Right-click and select the  **Design Function** from the popup menu or simply double-click the function/procedure.
or
- Click the  **Design Function** from the object pane toolbar.
- Modify function/procedure properties on the appropriate tabs of the Function/Procedure Designer.
- Click  **Save As**.

Edit Function/Procedure

To edit the existing function/procedure (manage its definition etc)



- Select the function/procedure for editing in the navigation pane/object pane.
- Right-click and select the  **Design Function** from the popup menu or simply double-click the function/procedure.
or
- Click the  **Design Function** from the object pane toolbar.
- Edit function/procedure properties on the appropriate tabs of the Function/Procedure Designer.

To change the name of the function/procedure


- Select the function/procedure for editing in the navigation pane/object pane.
- Right-click and select the **Rename** from the popup menu.

Run Function/Procedure

To run a function/procedure in the navigation pane/object pane



- Select the function/procedure for executing in the navigation pane/object pane.
- Click the  **Execute Function** from the object pane toolbar.
or
- Right-click and select  **Execute Function** from the popup menu.
- View/edit the returned data on the Result tab.

To run a function/procedure in the Function/Procedure Designer

- Create a new function/procedure/open the existing function/procedure.
- Click  **Run**.
- View/edit the returned data on the Result tab.

Delete Function/Procedure

To delete a function/procedure


- Select the function/procedure for deleting in the navigation pane/object pane.
- Right-click and select the  **Delete Function** from the popup menu.
or
- Click the  **Delete Function** from the object pane toolbar.
- Confirm deleting in the dialog window.

Achieve Function/Procedure Information

To achieve a function/procedure information

- Select the function/procedure in the navigation pane/object pane.
- Right-click the selected function/procedure and choose **Object Information** from the popup menu.
or
- Choose View -> Object Information in the main menu.

MySQL Function Wizard

Click the  **New Function** from the object pane toolbar. The **Function Wizard** will pop up and it allows you to create a procedure/function easily.

- [Setting Routine Type](#)
- [Setting Parameters for Procedure/Function](#)
- [Setting Return Type for Function](#)

You are allowed not to show the **Function Wizard** when create new procedure/function.

Hint: Once uncheck the **Show wizard next time**, you can go to Options to enable it.

Setting MySQL Routine Type

Select the type of the routine: **Procedure** or **Function**

Setting Parameters for MySQL Procedure/Function

Procedure

Define the parameter(s) of the procedure. Set the parameter **Mode**, **Name** and **Type** under corresponding columns.

Function

Define the parameter(s) of the function. Set the parameter **Name** and **Type** under corresponding columns.

Setting Return Type for MySQL Function

Select the **Return Type** from the list and enter the corresponding information: **Length**, **Decimals**, **Character Set** and/or **Enum**.

Note: Only function supports return type.

MySQL Function/Procedure Designer

Function/Procedure Designer is the basic Navicat tool for working with functions/procedures. It allows you to create new function/procedure and edit the existing function/procedure definition.

- [Editing Function/Procedure Definition](#)
- [Setting Advanced Function/Procedure Properties](#)
- Editing Function/Procedure Comment
- Function/Procedure SQL Preview
- [Viewing Function/Procedure Result](#)

Editing MySQL Function/Procedure Definition

Edit the function/procedure definition under the **Definition** tab. 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.

Example:

```
BEGIN
    RETURN CONCAT('Hello', name1, ' and ', name2, '!');
END
```

Hint: To customize the view of the editor and find out more features for sql editing, see Editor View and More 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**.

Setting Advanced MySQL Function/Procedure Properties

Security

The **SQL SECURITY** characteristic can be used to specify whether the routine should be executed using the permissions of the user who creates the routine or the user who invokes it. The default value is **Definer**.

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

Indicates 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

Indicates that the routine contains no SQL statements.

Reads SQL Data

Indicates that the routine contains statements that read data, but not statements that write data.


Modifies SQL Data

Indicates 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. The default is not deterministic.

Viewing MySQL Function/Procedure 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 popup.

Note: The **Result** tab displays the result data as grid.

Hint: Navicat supports to return 10 resultsets.



MySQL 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.

Just simply click  to open an object pane for **Event**. A right-click displays the popup menu or using the object pane toolbar, allowing you to create new, edit and delete the selected event.




Create Event

To create a new event

- Select anywhere on the object pane.
- Click the  **New Event** from the object pane toolbar.
or
- Right-click and select  **New Event** from the popup menu.
- Edit event properties on the appropriate tabs of the Event Designer.



Hint: To create new event you can also right-click the Event node of the navigation pane and select the  **New Event** from the popup menu.

To create a new event with modification as one of the existing event

- Select the event for modifying in the navigation pane/object pane.
- Right-click and select the  **Design Event** from the popup menu or simply double-click the event.
or
- Click the  **Design Event** from the object pane toolbar.
- Modify event properties on the appropriate tabs of the Event Designer.
- Click  **Save As**.

Edit Event

To edit the existing event (manage its definition etc)



- Select the event for editing in the navigation pane/object pane.
- Right-click and select the  **Design Event** from the popup menu or simply double-click the event.
or
- Click the  **Design Event** from the object pane toolbar.
- Edit event properties on the appropriate tabs of the Event Designer.

To change the name of the event

- Select the event for editing in the navigation pane/object pane.
- Right-click and select the **Rename** from the popup menu.

Delete Event

To delete an event

- Select the event for deleting in the navigation pane/object pane.
- Right-click and select the  **Delete Event** from the popup menu.
or
- Click the  **Delete Event** from the object pane toolbar.
- Confirm deleting in the dialog window.

Achieve Event Information

To achieve an event information

- Select the event in the navigation pane/object pane.
- Right-click the selected event and choose **Object Information** from the popup menu.
or
- Choose View -> Object Information in the main menu.

MySQL Event Designer

Event Designer is the basic Navicat tool for working with events. It allows you to create new event and edit the existing event definition.

- [Editing Event Definition](#)
- [Setting Advanced Event Properties](#)
- Editing Event Comment
- Event SQL Preview

Editing MySQL Event Definition

Edit the event definition under the **Definition** tab. 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 View and More Features.

Definer

Specifies 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.

Setting Advanced MySQL Event Properties

Edit the *ON SCHEDULE* clause under **Schedule** tab. The *ON SCHEDULE* clause determines when, how often, and for how long the SQL statement defined for the event repeats. This clause takes one of two forms:

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 Instance: ***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 <i>timestamp</i> must be in the future - you cannot schedule an event to take place in the past.
	<p>The <i>interval</i> portion consists of two parts, a quantity and a *unit of time.</p> <p>*YEAR QUARTER MONTH DAY HOUR MINUTE WEEK SECOND YEAR_MONTH DAY_HOUR DAY_MINUTE DAY_SECOND HOUR_MINUTE HOUR_SECOND MINUTE_SECOND</p>