

# Godot Database Manager

## Description:

**Godot Database Manager** is a plugin for **Godot Game Engine** (<https://godotengine.org/>) that can create local databases stored in JSON files.

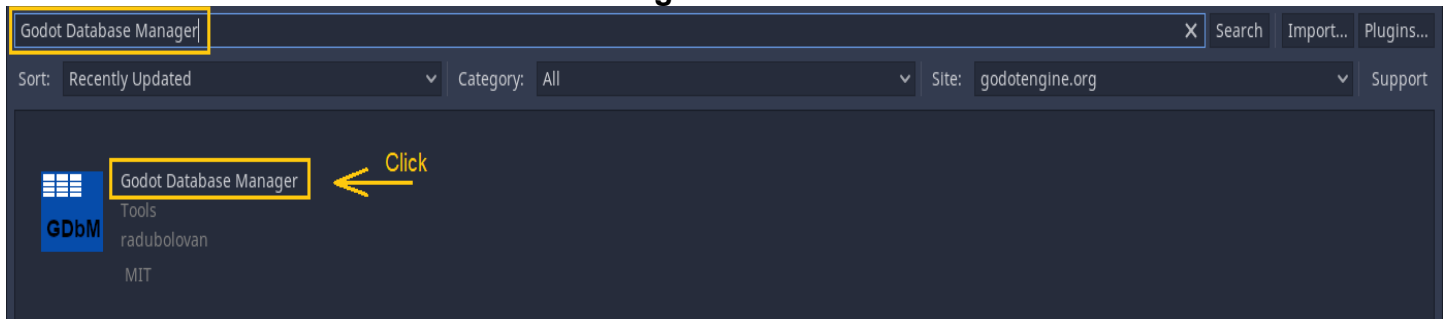
## Installation:

There are two ways to download and install the plugin into your project.

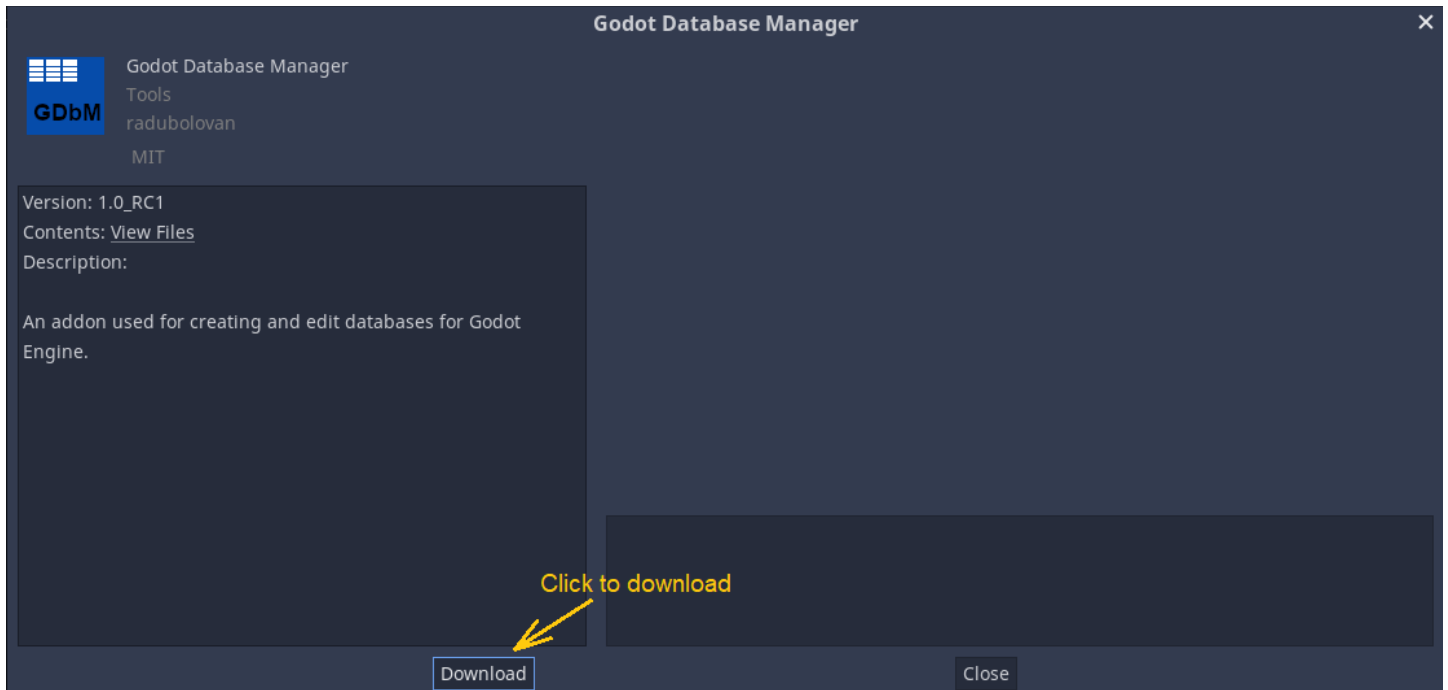
- 1) Directly in the **Godot Game Engine**'s editor through **Godot Asset Library**:
  - Open **Godot Game Engine** editor and access the AssetLib.



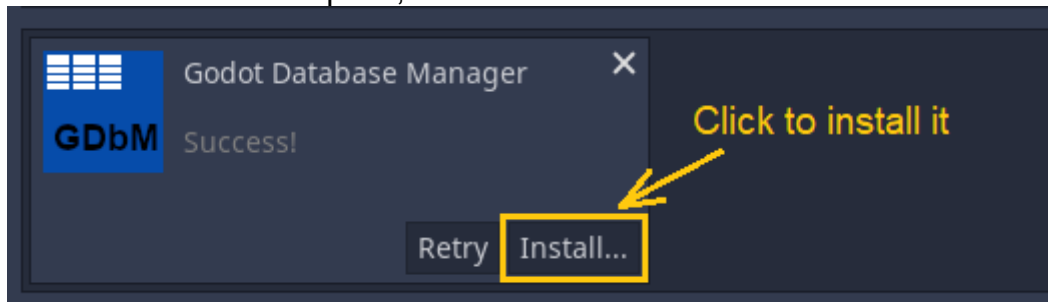
- Search for “**Godot Database Manager**” and click on it.



- Click on the “Download” button.

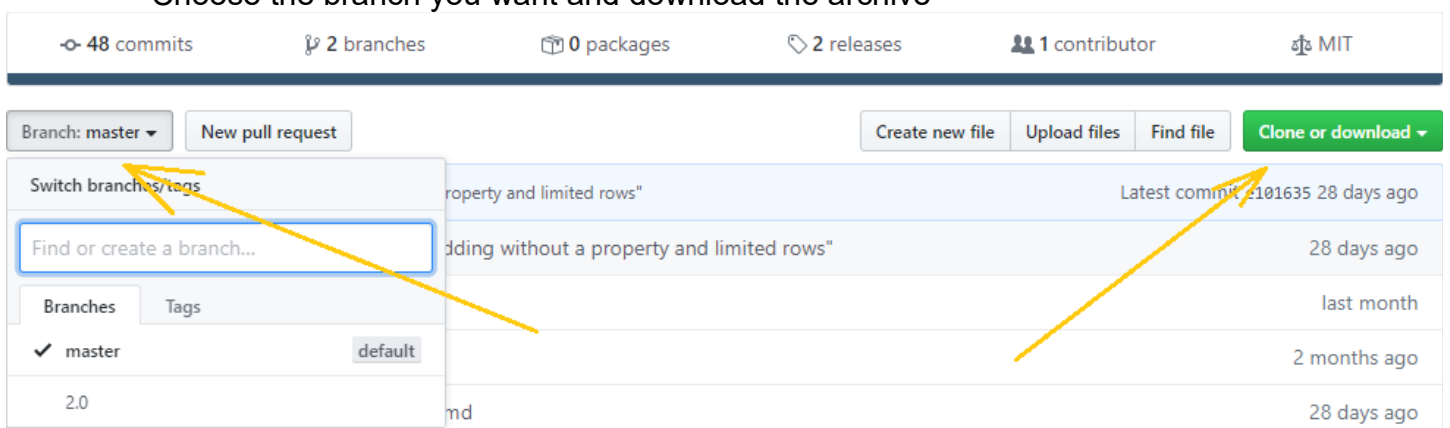


- After the download is complete, click on the “Install...” button.



## 2) Download it from Github

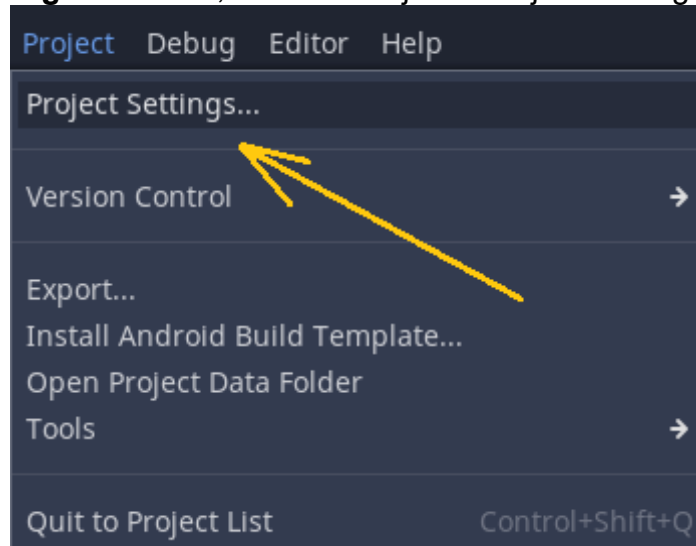
- You can download it from: <https://github.com/radubolovan/Godot-Database-Manager>
- Choose the branch you want and download the archive



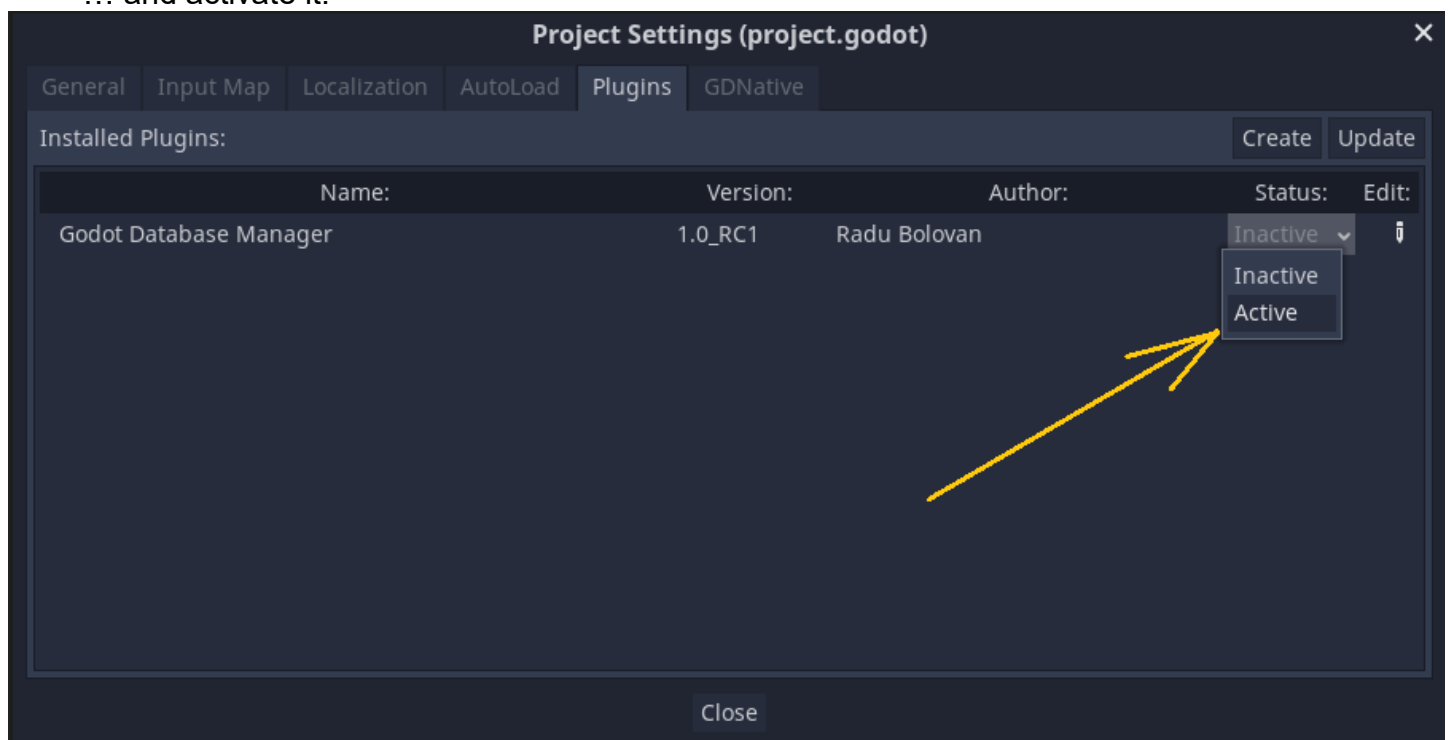
- Copy the "addons/godot\_db\_manager" directory (folder) into your project.

### Activation:

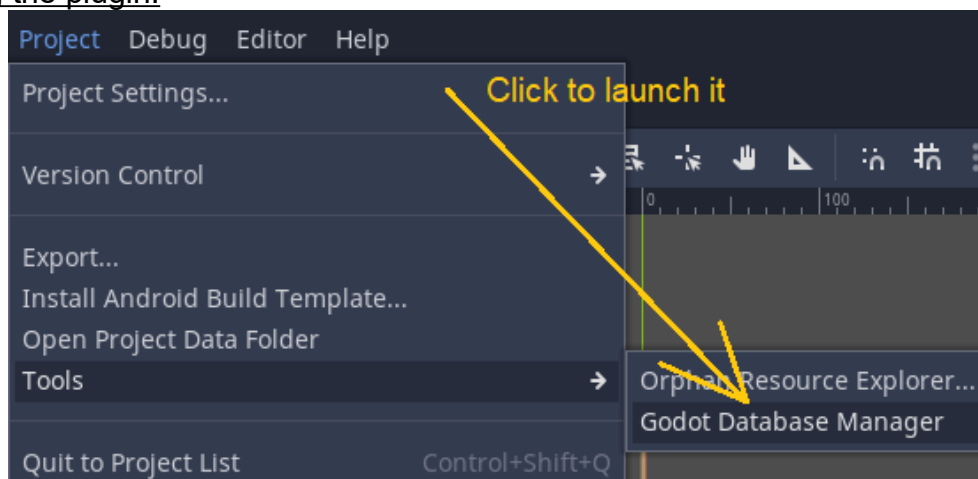
From **Godot Game Engine's** editor, access "Project->Project Settings..."



... and activate it.

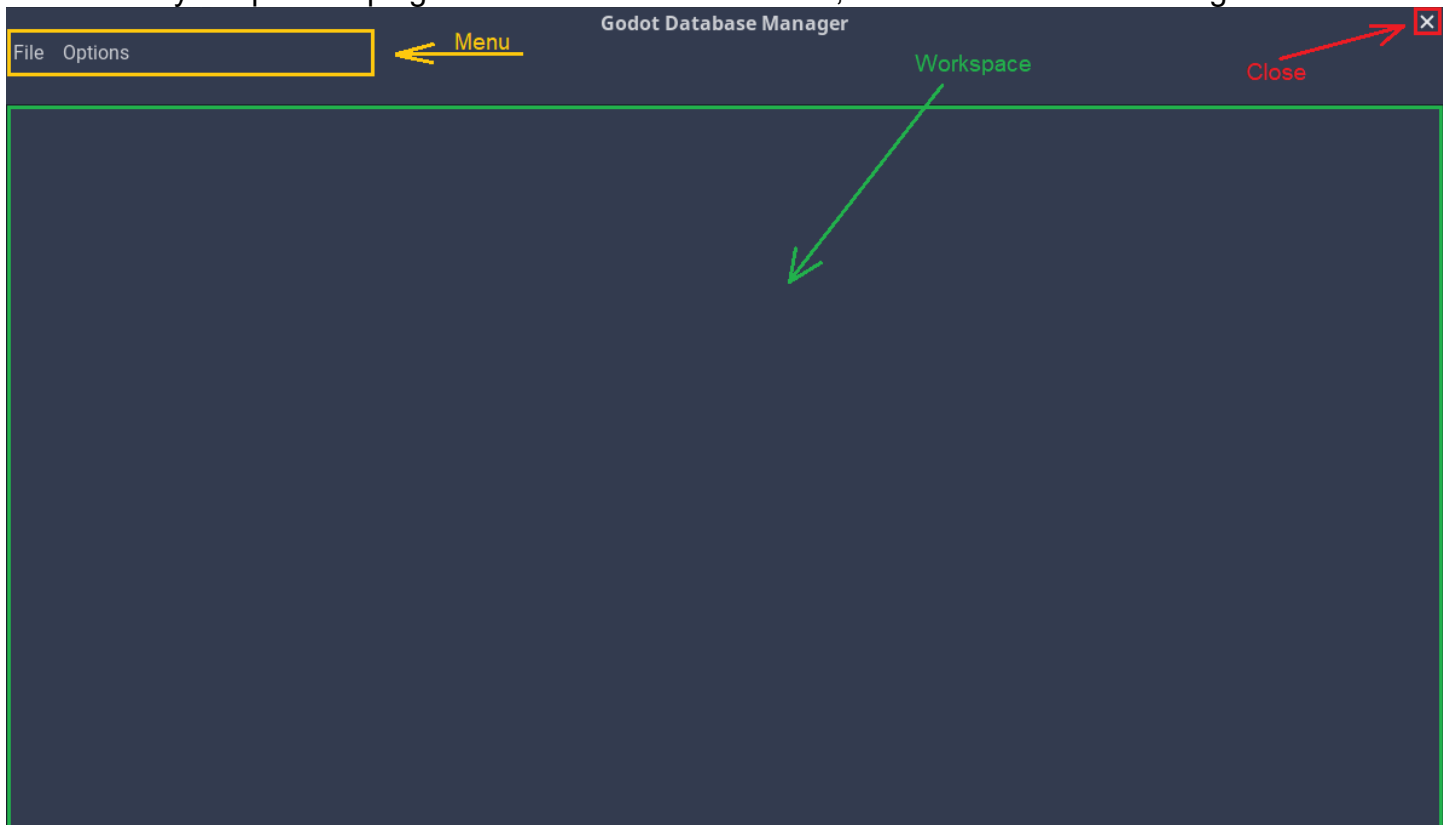


### Launching the plugin:



## The main interface:

When you open the plugin's interface for the first time, it should look like the image below.

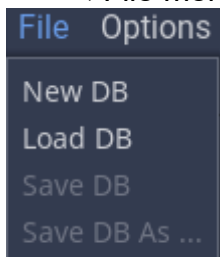


Close button

By clicking on it, will close the plugin's interface.

Menu

◆File menu



- "New DB": creates a new database
- "Load DB": loads a database from a JSON file
- "Save DB": saves the current database to a JSON file
- "Save DB As ...": saves the current database to a different JSON file

◆Options menu



- "Autosave on close": when enabled, all opened databases will be automatically saved.

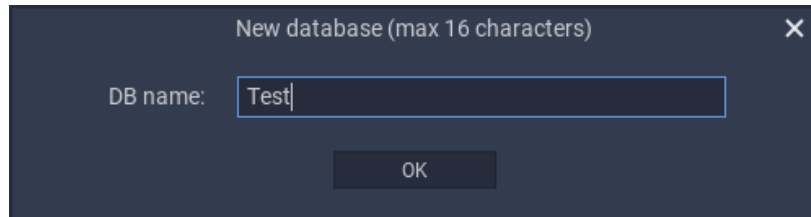
Workspace

It is the area where you will edit the databases, tables, properties and data.

### Creating a new database:

Choose "File -> New DB".

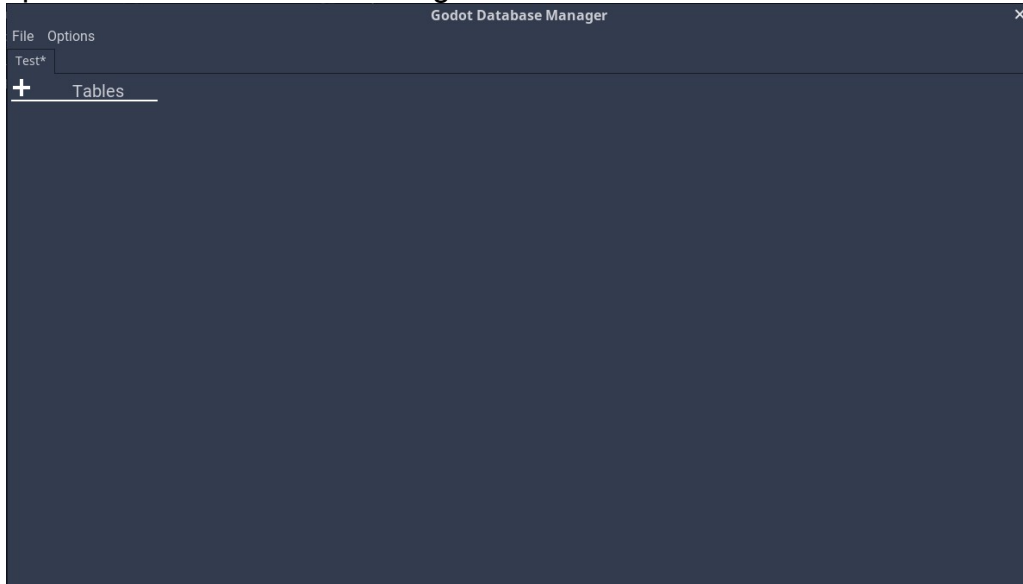
Type in the database name and click the "OK" button.



A database name cannot contain the following characters: "`~!@#\$\$%^&\*()=+[]{}|;: '\", < . > / ?".

Also there is a limit of 16 characters when choosing a database name.

The workspace should look like the image below:

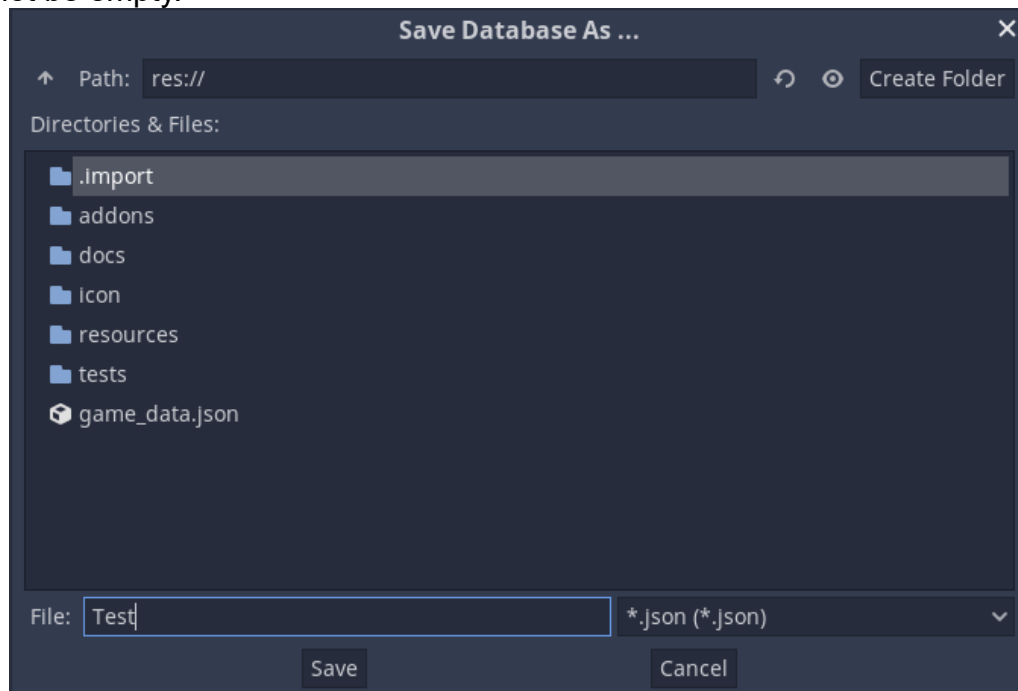


### Saving a database:

Choose "File menu -> Save DB".

Leave the file name as it is, or change it if you want to, and click the "Save" button.

The file name cannot contain the following characters: "`~!@#\$\$%^&\*()=+[]{}|;: '\", < . > / ?". Also the file name cannot be empty.



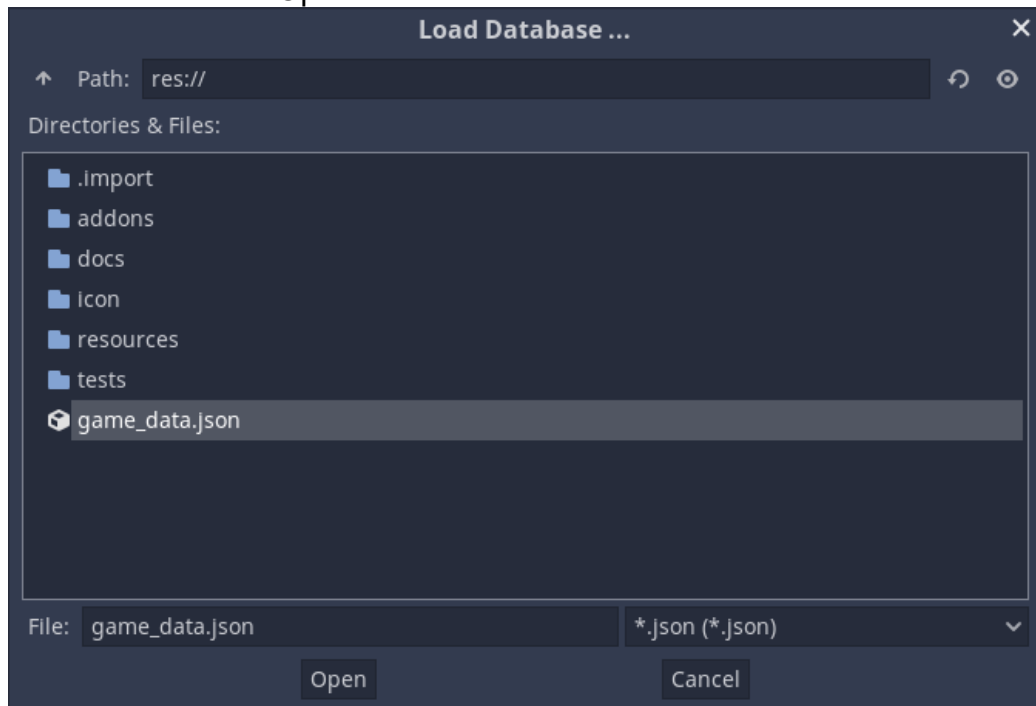
Once a database is saved to a file, that file will be associated to the database and next time when saving it, will not ask you to choose in which file you want to save it.

If you want to choose a different file, choose "File menu -> Save DB As ...".

### Loading a database:

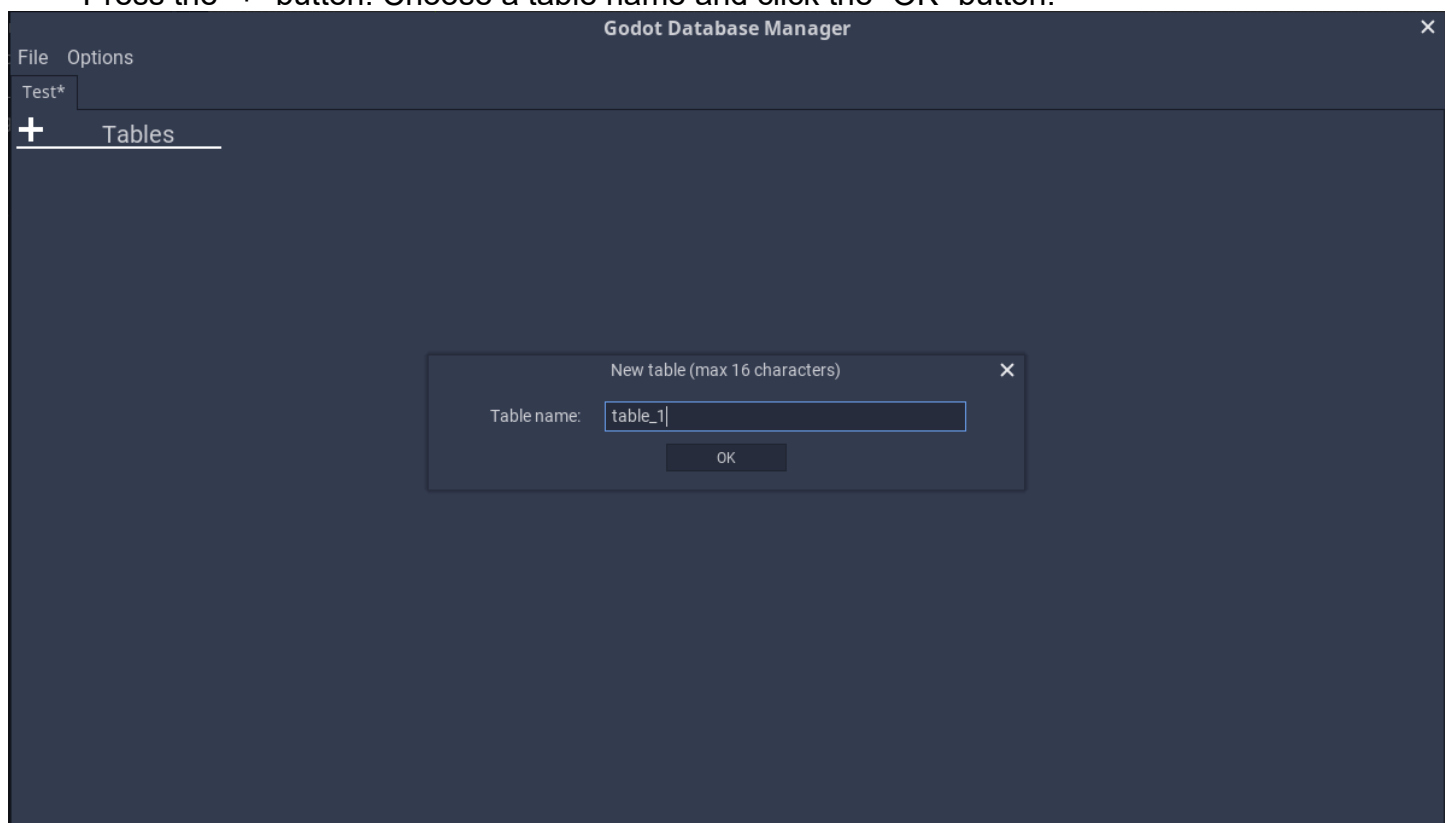
Choose “File menu -> Load DB”.

Choose a file and click the “Open” button.



### Creating a table and table editor interface:

Press the “+” button. Choose a table name and click the “OK” button.



There is a limit of 16 characters when choosing a table name.

OBS: the table name must be unique in the database.

After creating the table, the workspace should look like the image below:

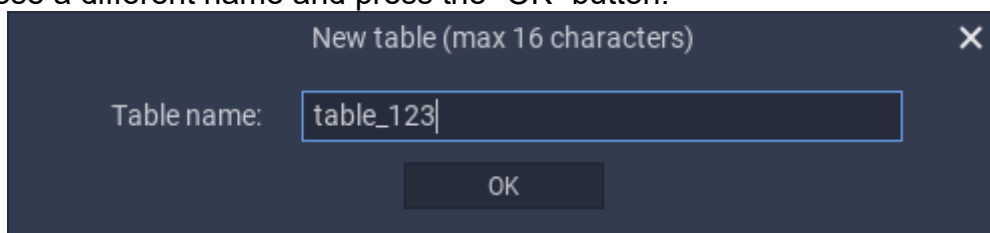


Renaming and deleting tables:

Click the “Edit Table” button. See the image below for more details:



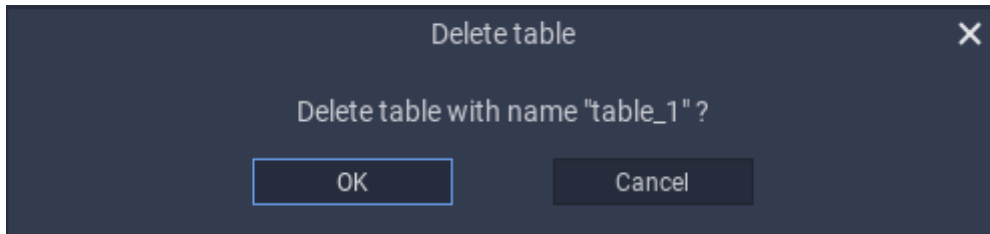
Then choose a different name and press the “OK” button.



Click the “Delete Table” button. See the image below for more details:



Then confirm the deletion of the table:



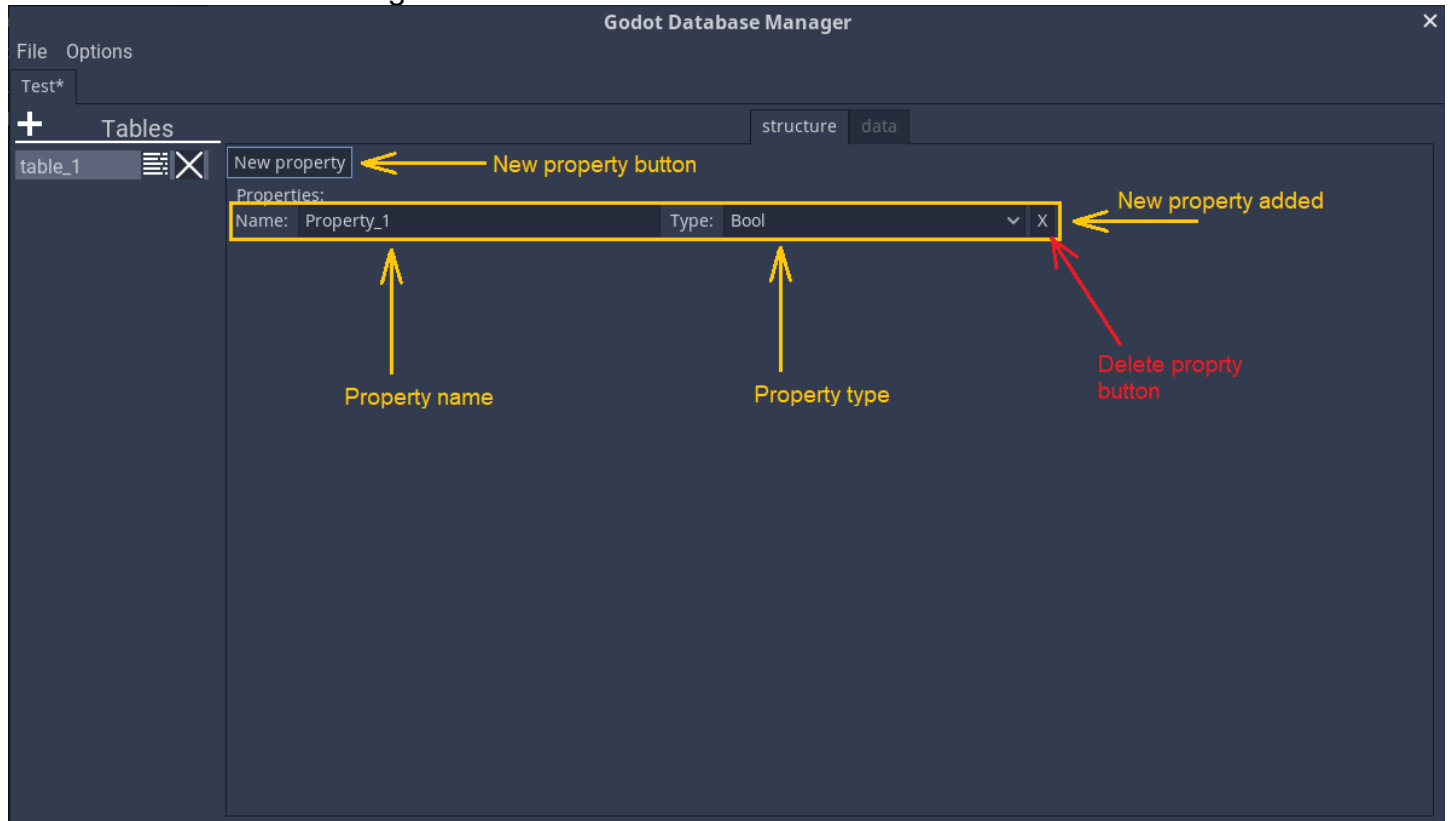
### Creating, editing and deleting the properties:

Click on the "New property" button and a new property will automatically be added in the table.

After that you can edit the property name and its type.

You can also delete a property by clicking the "Delete property" button.

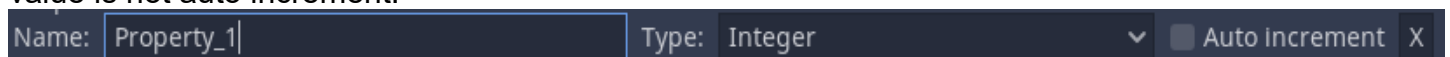
See details in the image below:



The type of the property can be:

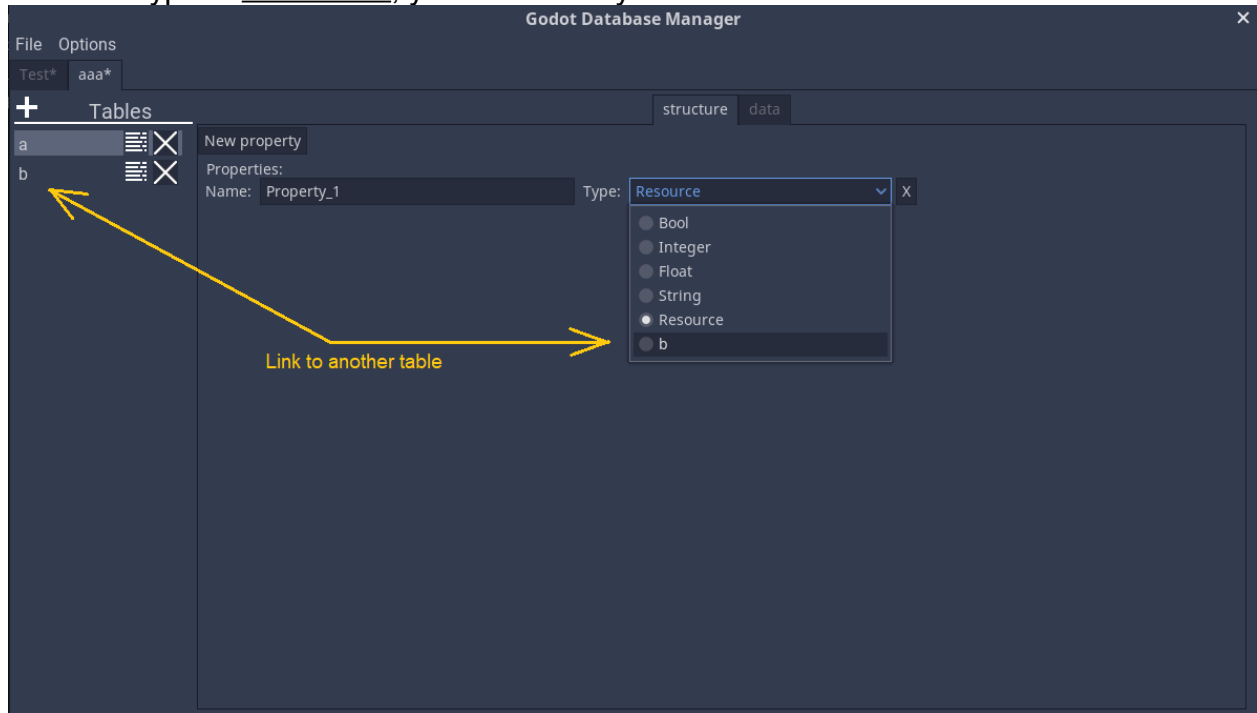
- **Boolean** (Bool): true or false.
- **Integer**: integer number.
- **Float**: floating point number.
- **String**: text.
- **Resource**: a link to a file that is a resource (text, image, sound, video, etc).
- **User data**: a link to another table from the database.

If the type is **Integer**, you can also choose the option to auto increment the data. The default value is not auto increment.



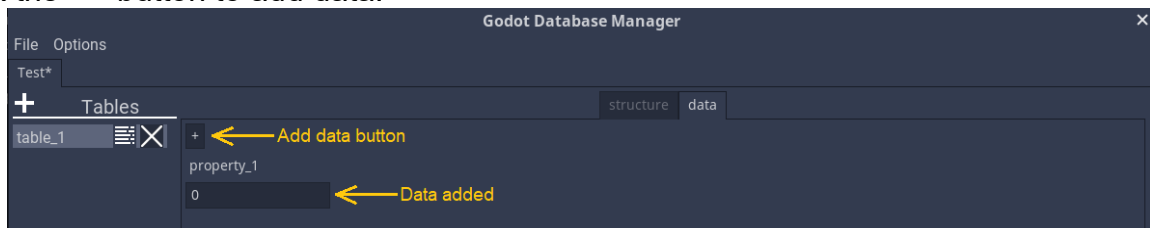


If the data type is **User Data**, you can directly choose select the table from the database.

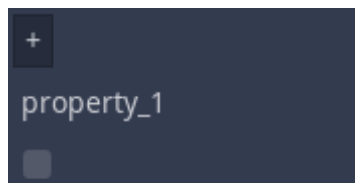


### Adding and edit data

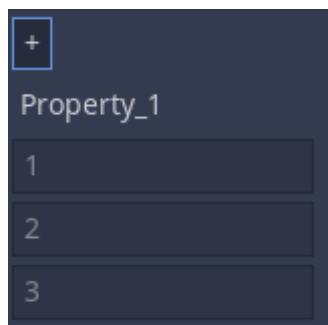
Click the “+” button to add data.



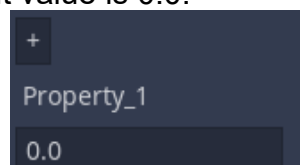
If the data type is **Boolean**, the data is represented by a checkbox and default is false (unchecked).



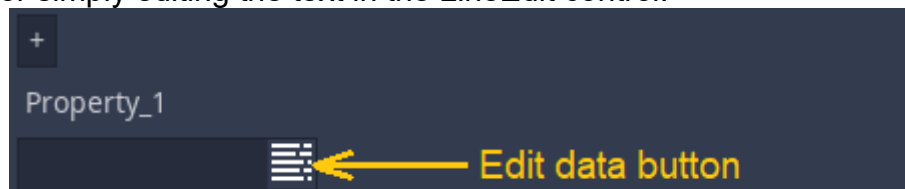
If the data type is **Integer**, the default value is 0. But if the auto increment option is set to true, the default value is 1 and the data cannot be edited. Also, if you add more data, the next values will be auto incremented.



If the data type is **Float**, the default value is 0.0.



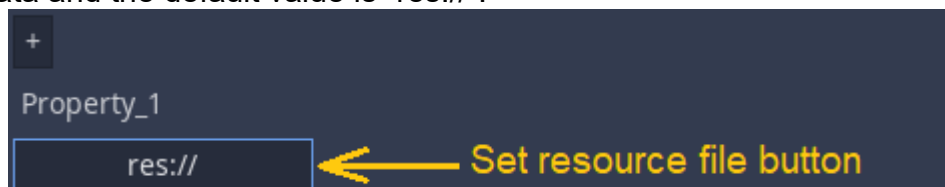
If the data type is **String**, the default data is "" (an empty string). The data can be edited via "Edit data" button or simply editing the text in the LineEdit control.



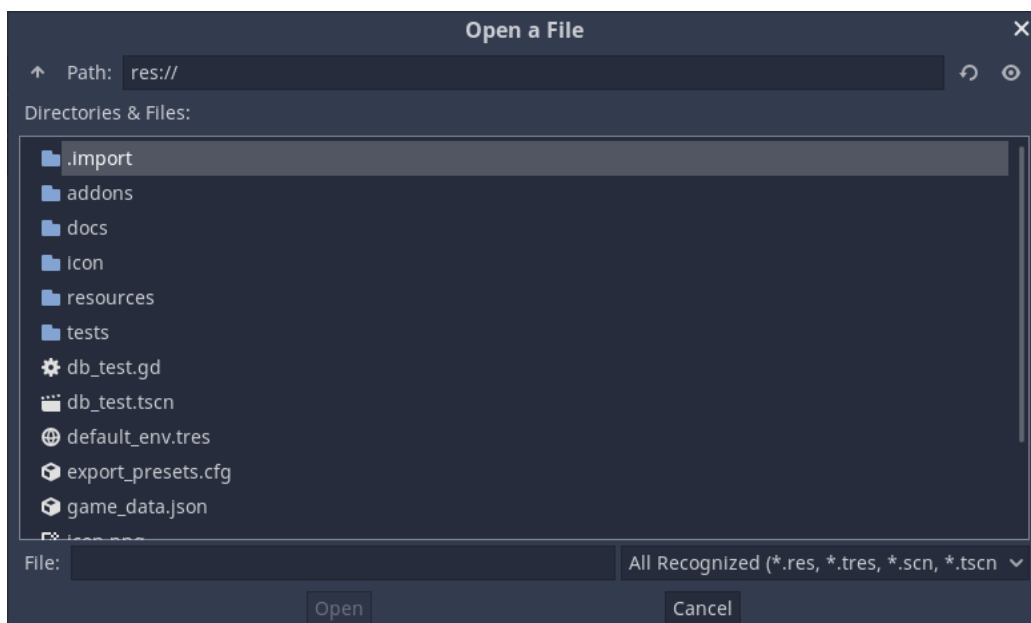
By pressing the "Edit Data" button, a text editor will show up letting you know to edit the data.



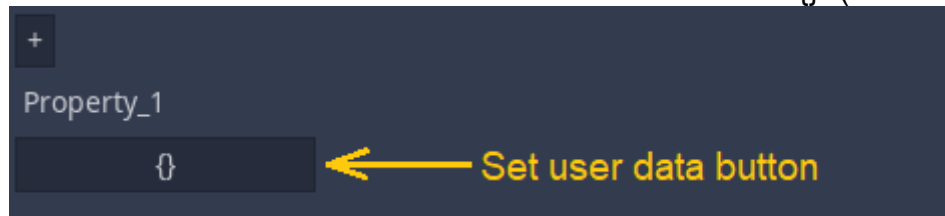
If the data type is **Resource**, editing the data is done via a button. The name of the button represents the data and the default value is "res://".



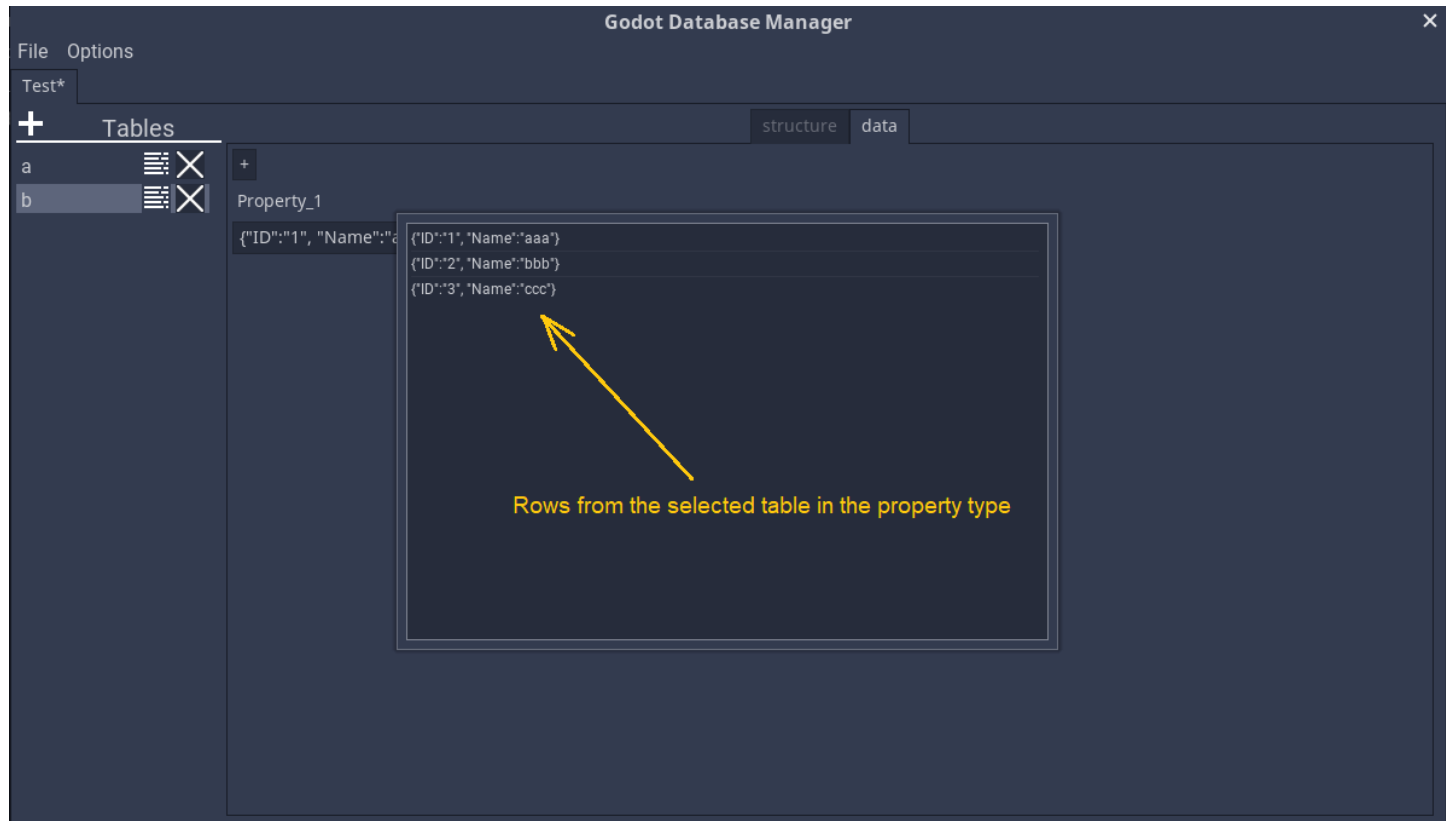
By clicking the button, a dialog will shown up letting you to choose a resource file from the project files.



If the data type is **User Data**, editing the data is done via a button. The name of the button represents a row of data from the selected table and the default value is “{}” (as a JSON).



By clicking the button, a popup will show up letting you to choose a row of data from the selected table.



# **GDScript files and classes**

## Constants

File: /addons/godot\_db\_manager/core/GDDDBConstants.gd

**c\_gdddb\_signature = "GDDB\_ver"**

- Godot Database Manager file signature.
- A constant for recognize the GDDDB files.
- Type: String

**c\_gdddb\_ver = "\_current\_version\_"**

- Current API version
- Type: String

**c\_invalid\_id = -1**

- A constant used to recognize an integer that is not initialized.
- Type: int

**c\_max\_db\_name\_len = 16**

- A constant that represents the maximum length of a database name.
- Type: int

**c\_max\_table\_name\_len = 16**

- A constant that represents the maximum length of a table name.
- Type: int

**c\_invalid\_characters = "~!@#\$\$%^&\*()=+[]{}\\|;:\"\<.>/?"**

- A constant containing all the invalid characters when naming a database.
- Type: String

**c\_addon\_main\_path = "res://addons/godot\_db\_manager/"**

- A constant used for easy access the plugin's path.
- Type: String

## Types

File: /addons/godot\_db\_manager/core/GDDBTypes.gd

### **Database loading errors.**

These are returned when loading a database.

**e\_db\_invalid\_file** – Loading file is not a valid GDDB database file.

**e\_db\_invalid\_ver** – The version of the loading database file is not the current one.

**e\_db\_valid** – The database has been loaded without errors.

### **Property types.**

These are used to recognize the property types.

**e\_prop\_type\_bool** – Boolean

**e\_prop\_type\_int** – Integer

**e\_prop\_type\_float** – Floating point

**e\_prop\_type\_string** – String

**e\_prop\_type\_resource** – Resource

**e\_prop\_types\_count** – Used for counting the property types

### **Data filters.**

Used when querying the database for data.

**e\_data\_filter\_equal** – the value of the data must be equal

**e\_data\_filter\_not\_equal** – the value of the data must be different

**e\_data\_filter\_less** – the value of the data must be less

**e\_data\_filter\_greater** – the value of the data must be greater

**e\_data\_filter\_lequal** – the value of the data must be less or equal

**e\_data\_filter\_gequal** – the value of the data must be greater or equal

## Global functions

File: /addons/godot\_db\_manager/core/GDDBGlobals.gd

### **get\_data\_name(data\_type : int) -> String**

Description: returns the name of the data type

– Arguments:

– **data\_type**: the type of the data. See **property type** enum for more info.

### **get\_data\_filter\_name(data\_filter\_type : int) -> String**

Description: returns the name of the data filter

– Arguments:

– **data\_filter\_type**: the type of the data filter. See **data filter** enum for more info.

### **check\_db\_name(db\_name : String) -> bool**

Description: checks name of a database. Returns **true** if the name of the database contains valid characters, otherwise **false**. See **c\_invalid\_characters** constant for more details.

– Arguments:

– **db\_name**: the name of the database

### **get\_json\_from\_row(table : Object, row\_idx : int) -> String**

Description: returns a json out of a row data from a table

– Arguments:

– **table**: the table in the database

– **row\_idx**: the index of the row in the table

### **get\_digits\_count(number : int) -> int**

Description: returns the count of the digits out of a number

– Arguments:

– **number**: an arbitrary number

### **handle\_string(text : String) -> String**

Description: replace special characters in a string to handle properly saving it into a database.

– Arguments:

– **text**: the original text

## Classes

### **GDDDBMan**

Description: Godot database manager

File: /addons/godot\_db\_manager/core/db\_man.gd

Extends: Object

Members:

**m\_databases** – array of databases.

Methods:

**add\_database(db\_name : String) -> int**

Description: adds a database. Returns the id of the database just inserted or **c\_invalid\_id** if the database cannot be added.

– Arguments:

–**db\_name**: the name of the database to be added. The name must be unique.

**load\_database(filepath : String) -> int**

Description: loads a database from a file. Returns the id of the database or an error code. See “Database Loading Errors” for more details.

– Arguments:

–**filepath**: the path of the database file.

**erase\_db\_at(idx : int) -> void**

Description: deletes a database at a given index.

– Arguments:

–**idx**: the index of the database.

**erase\_db\_by\_id(db\_id : int) -> void**

Description: deletes a database by an id.

– Arguments:

–**db\_id**: the id of the database.

**erase\_db\_by\_name(db\_name : String) -> void**

Description: deletes a database by a name.

– Arguments:

–**db\_name**: the name of the database.

**get\_databases\_count() -> int**

Description: returns the databases count.

### **get\_db\_at(idx : int) -> Object**

Description: returns a database at a given index.

– Arguments:

–**idx**: the index of the database.

### **get\_db\_by\_id(db\_id : int) -> Object**

Description: returns a database by an id.

– Arguments:

–**db\_id**: the id of the database.

### **get\_db\_by\_name(db\_name : String) -> Object**

Description: returns a database by a name.

– Arguments:

–**db\_name**: the name of the database.

### **generate\_new\_db\_id() -> int**

Description: generates a new database id. Internal usage.

### **can\_add\_db(db\_name : String) -> bool**

Description: checks if a database with the same name already exists. Returns **true** if the name is unique, otherwise **false**.

– Arguments:

–**db\_name**: the name of the database.

### **clear() -> void**

Description: clears all databases and the array.

### **dump(to\_console : bool) -> String**

Description: dumps the content of the database. Returns a string containing the dump.

– Arguments:

–**to\_console**: if true, the dump will also be added to the debug console.



## **GDDatabase**

Description: Database class.

File: /addons/godot\_db\_manager/core/database.gd

Extends: Object

Members:

**m\_db\_type : int** – database type. Currently only JSON type is supported.

**m\_db\_id : int** – database id.

**m\_db\_name : String** – database name.

**m\_tables** – array of tables

**m\_db\_filepath : String** – database filepath

**m\_is\_dirty : bool** – a flag to check if the database is modified. Used for the interface.

Methods:

**set\_db\_id(db\_id : int) -> void**

Description: sets the database id.

– Arguments:

–**db\_id**: the id of the database.

**get\_db\_id() -> int**

Description: returns the id of the database.

**set\_db\_name(db\_name : String) -> bool**

Description: sets the database name.

– Arguments:

–**db\_name**: the name of the database.

**get\_db\_name() -> String**

Description: returns the name of the database.

**set\_db\_filepath(filepath : String) -> void**

Description: sets the database filepath.

– Arguments:

–**filepath**: the filepath of the database.

**get\_db\_filepath() -> String**

Description: returns the filepath of the database.

### **can\_add\_table(table\_name : String, table\_id) -> bool**

Description: checks if a table can be added. Returns **true** if the name is unique and the table can be added, otherwise **false**.

– Arguments:

–**table\_name**: the name of the table.

–**table\_id**: the id of the table. This is used for renaming an existing table.

### **add\_table(table\_name : String) -> int**

Description: adds a table into the database. Returns the table id if the table is successfully added or **c\_invalid\_id** if a table with the same name already exists.

– Arguments:

–**table\_name**: the name of the table.

### **edit\_table\_name(table\_name : String, table\_id : int) -> bool**

Description: renames a table. If the name is used by another table returns **false**, otherwise **true**.

– Arguments:

–**table\_name**: the name of the table.

–**table\_id**: the id of the table.

### **delete\_table\_at(idx : int) -> void**

Description: deletes a table at a given index.

– Arguments:

–**idx**: the index of the table.

### **delete\_table\_by\_id(table\_id: int) -> void**

Description: deletes a table by an id.

– Arguments:

–**table\_id**: the id of the table.

### **delete\_table\_by\_name(table\_name: String) -> void**

Description: deletes a table by a name.

– Arguments:

–**table\_name**: the name of the table.

### **generate\_new\_table\_id() -> int**

Description: generates a new table id. Internal usage.

### **get\_tables\_count() -> int**

Description: returns the tables count

### **is\_table\_exists(table\_name : String) -> bool**

Description: returns **true** if a table with the name exists, otherwise **false**.

– Arguments:

–**table\_name**: the name of the table.

### **get\_table\_at(idx: int) -> Object**

Description: returns a table at a given index.

– Arguments:

–**idx**: the index of the table.

### **get\_table\_by\_id(table\_id: int) -> Object**

Description: returns a table by an id.

– Arguments:

–**table\_id**: the id of the table.

### **get\_table\_by\_name(table\_name: String) -> Object**

Description: returns a table by a name.

– Arguments:

–**table\_name**: the name of the table.

### **clear() -> void**

Description: clears the database content.

### **set\_dirty(dirty : bool) -> void**

Description: sets a flag if the database is modified.

– Arguments:

–**dirty**: if true, then the database is modified.

### **is\_dirty() -> bool**

Description: returns **true** if a database has been modified, otherwise **false**.

### **save\_db() -> void**

Description: saves a database.

### **load\_db() -> int**

Description: loads a database. Returns `e_db_valid` if the database is successfully loaded or an error code. See “Database Loading Errors” for more details.

### **dump() -> String**

Description: dumps the content of the database. Returns a string containing the dump.

## **GDDDBTable**

Description: Table class.

File: /addons/godot\_db\_manager/core/db\_table.gd

Extends: Object

Members:

**m\_table\_id : int** – table id

**m\_table\_name : String** – table name

**m\_props** – array of properties

**m\_data** – array of data

**m\_rows\_count : int** – rows count

**m\_parent\_database : Object** – parent database

Methods:

**set\_table\_id(table\_id : int) -> void**

Description: sets the table id.

– Arguments:

–**table\_id**: the id of the table.

**get\_table\_id() -> int**

Description: returns the id of the table.

**set\_table\_name(table\_name: String) -> void**

Description: sets the table name.

– Arguments:

–**table\_name**: the name of the table.

**get\_table\_name() -> String**

Description: returns the name of the table.

**set\_parent\_database(db : Object) -> void**

Description: sets the parent database.

– Arguments:

–**db**: database.

**get\_parent\_database() -> Object**

Description: returns the parent database.

**add\_prop(prop\_type : int, prop\_name : String) -> int**

Description: adds a property.

– Arguments:

–**prop\_type**: the type of the property.

–**prop\_name**: the name of the property.

**add\_table\_prop(prop\_name : String, table\_name : String) -> int**

Description: adds a property as a table type. Returns the property id.

– Arguments:

- prop\_name**: the name of the property.
- table\_name**: the name of the table.

**link\_tables\_props() -> void**

Description: links custom properties from tables. Internal usage.

**edit\_prop(prop\_id : int, prop\_type : int, prop\_name: String) -> void**

Description: edits a property.

– Arguments:

- prop\_id**: the id of the property.
- prop\_type**: the type of the property.
- prop\_name**: the name of the property.

**enable\_prop\_autoincrement(prop\_id : int, enable : bool) -> void**

Description: enables or disables auto increment option to a property.

– Arguments:

- prop\_id**: the id of the property.
- enable**: enable/disable flag.

**delete\_prop(prop\_id : int) -> void**

Description: deletes a property by an id.

– Arguments:

- prop\_id**: the id of the property.

**generate\_new\_prop\_id() -> int**

Description: generates a new property id. Internal usage.

**get\_props\_count() -> int**

Description: returns the properties count.

**get\_prop\_at(idx : int) -> Object**

Description: returns a property at a given index.

– Arguments:

- prop\_idx**: the index of the property.

### **get\_prop\_by\_id(prop\_id : int) -> Object**

Description: returns a property by an id.

– Arguments:

–**prop\_id**: the id of the property.

### **add\_blank\_row() -> void**

Description: adds a blank row of data.

### **add\_row(data\_array : Array) -> void**

Description: adds a row of data.

– Arguments:

–**data\_array**: the array with data.

### **remove\_row(row\_idx : int) -> void**

Description: deletes a row of data at a given index.

– Arguments:

–**row\_idx**: the index of the row.

### **get\_rows\_count() -> int**

Description: returns the rows count.

### **edit\_data(prop\_id : int, row\_idx : int, data : String) -> void**

Description: edits a data.

– Arguments:

–**prop\_id**: the id of the property.

–**row\_idx**: the index of the row.

–**data**: the new data.

### **edit\_data\_by\_prop\_name(prop\_name : String, row\_idx : int, data : String) -> void**

Description: edits a data.

– Arguments:

–**prop\_name**: the name of the property.

–**row\_idx**: the index of the row.

–**data**: the new data.

### **get\_data\_size() -> int**

Description: returns the amount of data.

### **get\_all\_data() -> Array**

Description: returns the data array.

**get\_data\_at(idx : int) -> String**

Description: returns a data by a given index.

– Arguments:

–**idx**: the index of the data.

**get\_dictionary\_at(idx : int) -> Dictionary**

Description: returns a dictionary containing data by a given index.

– Arguments:

–**idx**: the index of the data.

**get\_data(prop\_id : int, row\_idx : int) -> String**

Description: returns a data by a property id and at a given row index.

– Arguments:

–**prop\_id**: the id of the property.

–**row\_idx**: the index of the row.

**get\_dictionary(prop\_id : int, row\_idx : int) -> Dictionary**

Description: returns a dictionary containing the data by a property id and a row index.

– Arguments:

–**prop\_id**: the id of the property.

–**row\_idx**: the index of the row.

**get\_data\_at\_row\_idx(row\_idx : int) -> Array**

Description: returns an array of data at a given row index.

– Arguments:

–**row\_idx**: the index of the row.

**get\_dictionary\_at\_row\_idx(row\_idx : int) -> Dictionary**

Description: returns a dictionary of data at row index.

– Arguments:

–**row\_idx**: the index of the row.

**get\_data\_by\_prop\_id(prop\_id : int, data\_filter : int) -> Array**

Description: returns an array of data by a property id and using a filter.

– Arguments:

–**prop\_id**: the id of the property.

–**data\_filter**: the filter type of the data. See data filters for more details.

### **get\_data\_by\_prop\_name(prop\_name : String) -> Array**

Description: returns an array of data by a property name

– Arguments:

–**prop\_name**: the name of the property.

### **get\_data\_by\_data(data\_value : String, data\_filter : int) -> Array**

Description: returns an array of data filtered by data and data filter type.

– Arguments:

–**data\_value**: the value of the data.

–**data\_filter**: the data filter type. See data filters for more details.

### **get\_data\_by\_prop\_name\_and\_data(prop\_name : String, data\_value : String) -> Array**

Description: returns an array of data filtered by a property name and a data value.

– Arguments:

–**prop\_name**: the name of the property.

–**data\_value**: the value of the data.

– Example:

–get\_data\_by\_prop\_name\_and\_data("id", "1") => will return a row of data where the "id" is "1".

### **get\_dictionary\_by\_prop\_name\_and\_data(prop\_name : String, data\_value : String) -> Array**

Description: returns a dictionary of data by a property name and a data value.

– Arguments:

–**prop\_name**: the name of the property.

–**data\_value**: the value of the data.

### **clear() -> void**

Description: clears the table's content.

### **dump() -> String**

Description: dumps the content of the table. Returns a string containing the dump.



## **GDDDBProperty**

Description: Property class.

File: /addons/godot\_db\_manager/core/db\_prop.gd

Extends: Object

Members:

**m\_prop\_id : int** – property id

**m\_prop\_type : int** – property type. See property types for more details.

**m\_custom\_type : String** – custom property type. The name of another table in the database.

**m\_prop\_name : String** – property name.

**m\_autoincrement : bool** – auto increment option.

Methods:

**set\_prop\_id(prop\_id : int) -> void**

Description: sets the property id.

– Arguments:

–**prop\_id**: the id of the property.

**get\_prop\_id() -> int**

Description: returns the id of the property.

**set\_prop\_type(prop\_type : int) -> void**

Description: sets the property type.

– Arguments:

–**prop\_type**: the type of the property.

**get\_prop\_type() -> int**

Description: returns the type of the property.

**set\_prop\_custom\_type(prop\_type : String) -> void**

Description: sets the custom property type.

– Arguments:

–**prop\_type**: the name of the table.

**get\_prop\_custom\_type() -> String**

Description: returns the name of the custom property (another table's name).

**set\_prop\_name(prop\_name : String) -> void**

Description: sets the name of the property.

– Arguments:

–**prop\_name**: the name of the property.

### **get\_prop\_name() -> String**

Description: returns the name of the property.

### **enable\_autoincrement(enable : bool) -> void**

Description: enables / disables the auto increment option.

– Arguments:

–**enable**: enable/disable flag.

### **has\_autoincrement() -> bool**

Description: returns **true** if the property has the auto increment option enabled, otherwise **false**.

### **dump() -> String**

Description: dumps the content of the property. Returns a string containing the dump.

## **GDDbData**

Description: Data class.

File: /addons/godot\_db\_manager/core/db\_data.gd

Extends: Object

Members:

**m\_prop\_id : int** – property id

**m\_row\_idx : int** – row index.

**m\_data : String** – the data.

Methods:

### **set\_prop\_id(prop\_id : int) -> void**

Description: sets the property id.

– Arguments:

–**prop\_id**: the id of the property.

### **get\_prop\_id() -> int**

Description: returns the id of the property.

### **set\_row\_idx(row\_idx : int) -> void**

Description: sets the row index.

– Arguments:

–**row\_idx**: the index of the row.

### **get\_row\_idx() -> int**

Description: returns the index of the row.

**set\_data(data : String) -> void**

Description: sets the data.

– Arguments:

–**data**: the data.

**get\_data() -> String**

Description: returns the data.

**dump() -> String**

Description: dumps the content of the data. Returns a string containing the dump.