

CASE-SWDEP

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

v2.0

<https://github.com/cantario/CASE-SWDEP>

Глава 1	Начало работы с CASE-SWDEP	3
1.1	Введение в CASE-SWDEP	3
1.2	Требования к аппаратному обеспечению	3
1.3	Требования к программному обеспечению	3
1.4	Доступ к последней версии	3
1.5	Доступ к исходным текстам	4
1.6	Сборка программы из исходных текстов	5
1.6.2	Получение исходного кода	5
1.6.3	Сборка при помощи QtCreator	5
1.6.4	Сборка при помощи Qmake	6
1.7	Запуск программы из архива	6
1.8	Процедура создания плагинов	6
Глава 2	Создание простейшего SQL CREATE запроса	14
2.1	Пользовательский интерфейс	14
2.5	Подключение к БД и выполнение скрипта из программы	21
2.6	Структура концептуальной модели базы данных в формате CDMOD	22
2.7	Сохранение файла проекта	22
2.8	Открытие файла проекта	22
2.9	Изменение размера рабочей области	23

1.1 Введение в CASE-SWDEP

CASE-SWDEP – это система проектирования базы данных, разработанная на основе фреймворка Qt. CASE-SWDEP позволяет представить сущности и атрибуты предметной области в виде концептуальной модели данных и на основе этой модели создать саму базу данных.

CASE-SWDEP позволяет сгенерировать SQL запрос на создание базы данных при поддержке языка описания различных СУБД, посредством подключения соответствующих плагинов по мере необходимости. CASE-SWDEP предоставляет возможность сохранения концептуальной модели в формат CDMOD. Файлы данного формата доступны на чтение и редактирование в любом текстовом редакторе.

1.2 Требования к аппаратному обеспечению

Для CASE-SWDEP требуется 64-битный или 32-битный процессор и не менее 30 мб свободного места на жестком диске. Рекомендуемый объем оперативной памяти – 2гб. Также в качестве среды разработки является необходимым использование Qt версии не старше 5.14.

1.3 Требования к программному обеспечению

Для сборки CASE-SWDEP требуется фреймворк Qt версией не ниже 4.11 с основной Qt версии не ниже 5.14 с компилятором GCC разрядности 32 или 64 бит.

Текущая версия CASE-SWDEP поддерживает наиболее распространенные операционные системы:

- Windows 7/8/10 32bit/64bit
- Ubuntu Linux 32bit/64bit

1.4 Доступ к последней версии

Последняя версия программного средства CASE-SWDEP доступна по адресу <https://github.com/cantario/CASE-SWDEP/releases>

1.5 Доступ к исходным текстам

Последняя версия исходного кода CASE-SWDEP доступна по адресу <https://github.com/cantario/CASE-SWDEP>

Исходный код доступен для скачивания и компиляции, но он может содержать новые, неизданные функции и ошибки, поскольку находится на стадии разработки. Вся информация, необходимая для успешной сборки проекта находится в следующем пункте данного руководства.

1.6 Сборка программы из исходных текстов

1.6.1 Требования

- Qt 5.14+
- QtCreator 4.11+
- gcc 7.4+

1.6.2 Получение исходного кода

Клонировать исходный код из репозитория:

```
git clone https://github.com/cantario/CASE-SWDEP.git
```

или скачать архив. Архив распаковать в папку CASE-SWDEP.

Рекомендация: полный путь к папке CASE-SWDEP не должен содержать русские символы и пробелы.

1.6.3 Сборка при помощи QtCreator

Открыть файл «CASE_SWDEP.pro» в корневой папке проекта при помощи программы QtCreator. Далее нажать кнопку «Собрать» (ctrl + B) для сборки всего проекта, включая все подпроекты плагинов.

При необходимости сборки определенного подпроекта нужно перейти в папку данного подпроекта в окне навигации «Проекты» и выбрать пункт меню «Сборка – Пересобрать подпроект»

1.6.4 Сборка при помощи Qmake

Запустить терминал в корневой папке проекта. Набрать команды:

```
qmake [TODO]
```

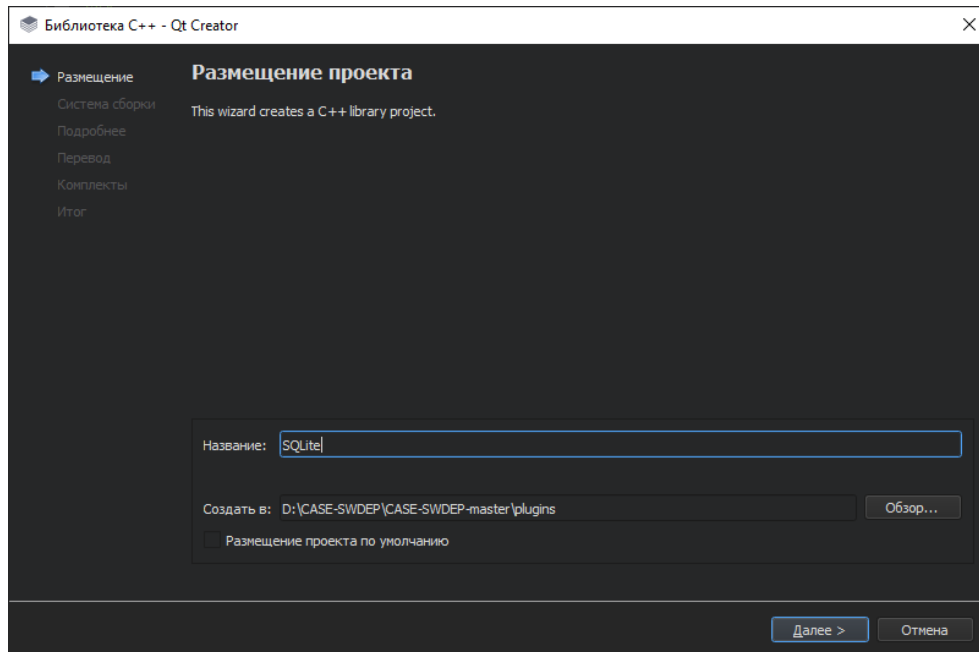
```
make
```

1.7 Запуск программы из архива

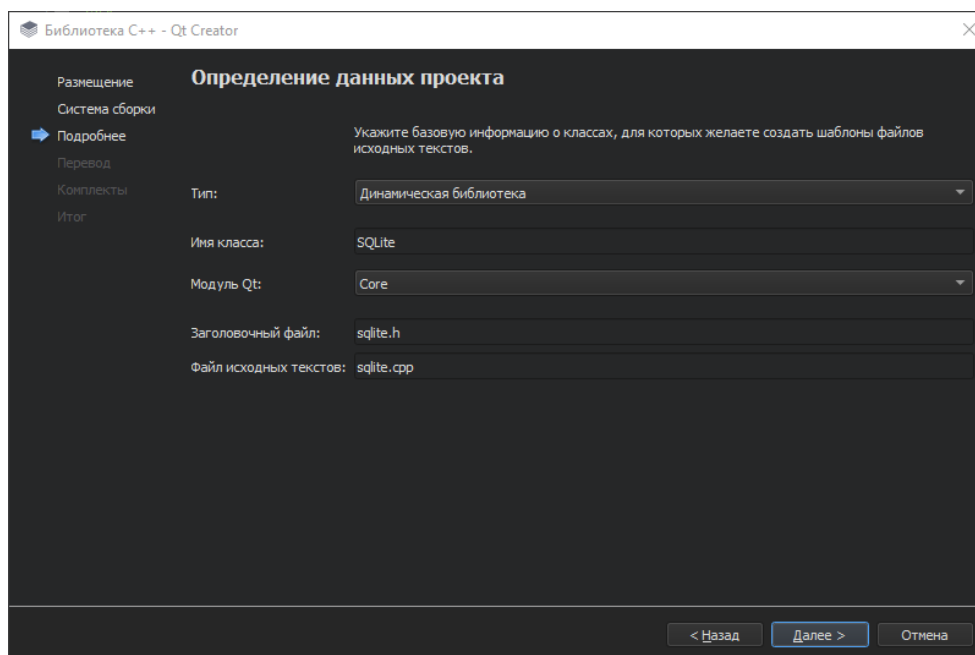
[TODO]

1.8 Процедура создания плагинов

Создание плагинов рассматривается в Qt версии 4.12.4. Чтобы создать плагин, необходимо создать проект «Библиотека C++». Пишем название плагина и выбираем его расположение. На рисунке представлено окно размещения проекта.



Выбираем систему сборки qmake -> тип библиотеки – динамическая библиотека -> модуль Qt – Core. Далее пишем название класса, для удобства назвать так же, как и проект. Информация о классе (см. рисунок ниже).



Выбираем комплект сборки Desktop Qt 5.15.0 MinGW 64-bit.

После нажатия кнопки «Завершить» появляются три файла: `sqlite.h`, `sqlite_global.h`, `sqlite.cpp` и файл проекта `SQLite.pro`. Файл `sqlite_global.h` содержит внешний интерфейс плагина. Удаляем его, так как в ПС был реализован общий интерфейс для всех модулей. Сначала требуется изменить файл проекта. Для того чтобы библиотека могла конвертироваться в файл DLL для ОС Windows и PLUGIN для ОС Linux, необходимо добавить флаг `DLLDESTDIR` и `DESTDIR` соответственно и указать путь хранения динамических библиотек. После компиляции проекта готовая библиотека будет создана в указанной папке. Так же следует добавить флаг `INCLUDEPATH` и указать расположение интерфейса для плагинов.


```

#-----
#
# Project created by QtCreator 2015-04-20T16:15:39
#
#-----

QT      -= gui

TARGET = SQLite3_35_5
TEMPLATE = lib

INCLUDEPATH += ../../CASE_SW

SOURCES += \
    sqlite3_35_5.cpp

HEADERS += \
    sqlite3_35_5.h

unix {
    CONFIG += plugin
    DESTDIR = $$PWD/../../build/case/lib/
}

win32 {
    CONFIG += dll
    DLLDESTDIR = $$PWD/../../build/case/lib/
}

```

Далее следует изменить заголовочный файл `sqlite.h`. Для начала в нём следует подключить файл интерфейса `cplugininterface.h`, чтобы плагин мог взаимодействовать с интерфейсом. Перед названием класса стоит опция «название класса» `+SHARED_EXPORT`. Эту опцию нужно убрать, так как она не используется. После этого наследуем класс плагина от класса интерфейса `CPluginInterface`. Далее внутри класса необходимо установить следующие макросы: `Q_OBJECT`, `Q_PLUGIN_METADATA`, `Q_INTERFACES`. Стоит уточнить

параметры двух последних макросов. Полная запись макроса `Q_PLUGIN_METADATA` выглядит так: `Q_PLUGIN_METADATA(IID "CPluginInterface/Plugin/1.0")`, где мы указываем внешний интерфейс в котором содержатся метаданные для плагина. В макросе `Q_INTERFACES` указывается класс внешнего интерфейса в котором присутствует макрос `Q_DECLARE_INTERFACE` сообщающий метаобъектной системе Qt об интерфейсе. Запись выглядит следующим образом: `Q_INTERFACES(DBPluginInterface)`.

После объявления макросов следует объявить основные функции для взаимодействия с плагином. В основной набор функций входят такие функции как: `QString getVersion()` – получение версии плагина; `QString getName()` – получение имени плагина; `QString query(IDataModel *dataModel) override` – генерация скрипта. Далее на рисунке приведён листинг заголовочного файла плагина.

```

#ifndef SQLITE3_35_5_H
#define SQLITE3_35_5_H

#include <cplugininterface.h>

class SQLite3_35_5 : public CPluginInterface
{
    Q_OBJECT
    Q_PLUGIN_METADATA(IID "CPluginInterface/Plugin/1.0")
    Q_INTERFACES(CPluginInterface)

public:
    QString name() override;
    QString version() override;
    QString query(IDataModel *dataModel) override;
};

#endif // SQLITE3_35_5_H

```

Реализация вышеперечисленных функций. Функция getVersion() возвращает строку типа QString с версией плагина.

```

QString SQLite3_35_5::version()
{
    return "3.35.5";
}

```

Функция getName возвращает значение QString с названием плагина.

```

QString SQLite3_35_5::name()
{
    return "SQLite";
}

```

Функция `query(IDataModel *dataModel)` override принимает аргументом таблицы, хранимые во внутреннем представлении. На выходе функция возвращает строку, содержащую скрипт для данной СУБД.

```
QString SQLite3_35_5::query(IDataModel *dataModel)
{
    _dataModel = dataModel;
    QString script = "";

    foreach (CTable *table, _dataModel->tablesSortedByReference()) {
        _table = (ITable *)table;
        if(_table->foreignRows().size() != 0)
        {
            script += "PRAGMA foreign_keys = ON;\n\n";
            break;
        }
    }

    foreach (CTable *table, _dataModel->tablesSortedByReference()) {
        _table = (ITable *)table;
        QString tableBody = "";
        QString primaryKey = "";

        foreach (CRow *row, _table->rows()) {
            _row = (IRow *)row;
            QString constraint = "";
            if(_row->unique() && !(_row->primaryKey()))
                constraint += " UNIQUE";
            if(_row->notNull() && !(_row->primaryKey()))
                constraint += " NOT NULL";
            if(_row->primaryKey())
                primaryKey += QString("%1, ")
                    .arg(_row->name());
            tableBody += QString("%1 %2%3, ")
                .arg(_row->name())
                .arg(_row->typeAsString())
                .arg(constraint);
        }
    }
}
```

```

// TABLE | (FOREIGN ROW, ROW)
QMap<QString, QList<QPair<QString, QString>>> fRowsToTables;
foreach (CForeignRow *foreignRow, _table->foreignRows()) {
    _foreignRow = (IForeignRow *)foreignRow;
    tableBody += QString("%1 %2, ")
                .arg(_foreignRow->name())
                .arg(_foreignRow->typeAsString());
    if(_foreignRow->primaryKey())
        primaryKey += QString("%1, ")
                        .arg(_foreignRow->name());
    if(fRowsToTables.contains(_foreignRow->tableName()))
    {
        QPair<QString, QString> pair(_foreignRow->name()
                                     ,_foreignRow->tableName());
        QList<QPair<QString, QString>> list;
        list = fRowsToTables.value(_foreignRow->tableName());
        list.append(pair);
        fRowsToTables.insert(_foreignRow->tableName(), list);
    }
    else
    {
        QPair<QString, QString> pair(_foreignRow->name()
                                     ,_foreignRow->tableName());
        QList<QPair<QString, QString>> list;
        list.append(pair);
        fRowsToTables.insert(_foreignRow->tableName(), list);
    }
}
tableBody.remove(tableBody.size() - 2, 2);
if(primaryKey != "")
{
    primaryKey.remove(primaryKey.size() - 2, 2);
    tableBody += QString(", PRIMARY KEY (%1)")
                .arg(primaryKey);
}

```

```

    if(!fRowsToTables.isEmpty())
    {
        tableBody += ", ";
        for(int i = 0; i < fRowsToTables.size(); i++)
        {
            QString fRowList, tRowList;
            for(int j = 0; j < fRowsToTables.values().at(i).size(); j++)
            {
                fRowList += fRowsToTables.values().at(i).at(j).first + ", ";
                tRowList += fRowsToTables.values().at(i).at(j).second + ", ";
            }
            fRowList.remove(fRowList.size() - 2, 2);
            tRowList.remove(tRowList.size() - 2, 2);
            tableBody += QString(" FOREIGN KEY (%3)"
                                " REFERENCES %1 (%2)"
                                " ON DELETE RESTRICT"
                                " ON UPDATE RESTRICT, ")
                        .arg(fRowsToTables.keys().at(i))
                        .arg(tRowList)
                        .arg(fRowList);
        }
        tableBody.remove(tableBody.size() - 2, 2);
    }
    if(tableBody != "")
    {
        script += QString("CREATE TABLE IF NOT EXISTS %1 ("
                        "%2"
                        ");\n\n")
                .arg(_table->name())
                .arg(tableBody);
    }
    return script;
}

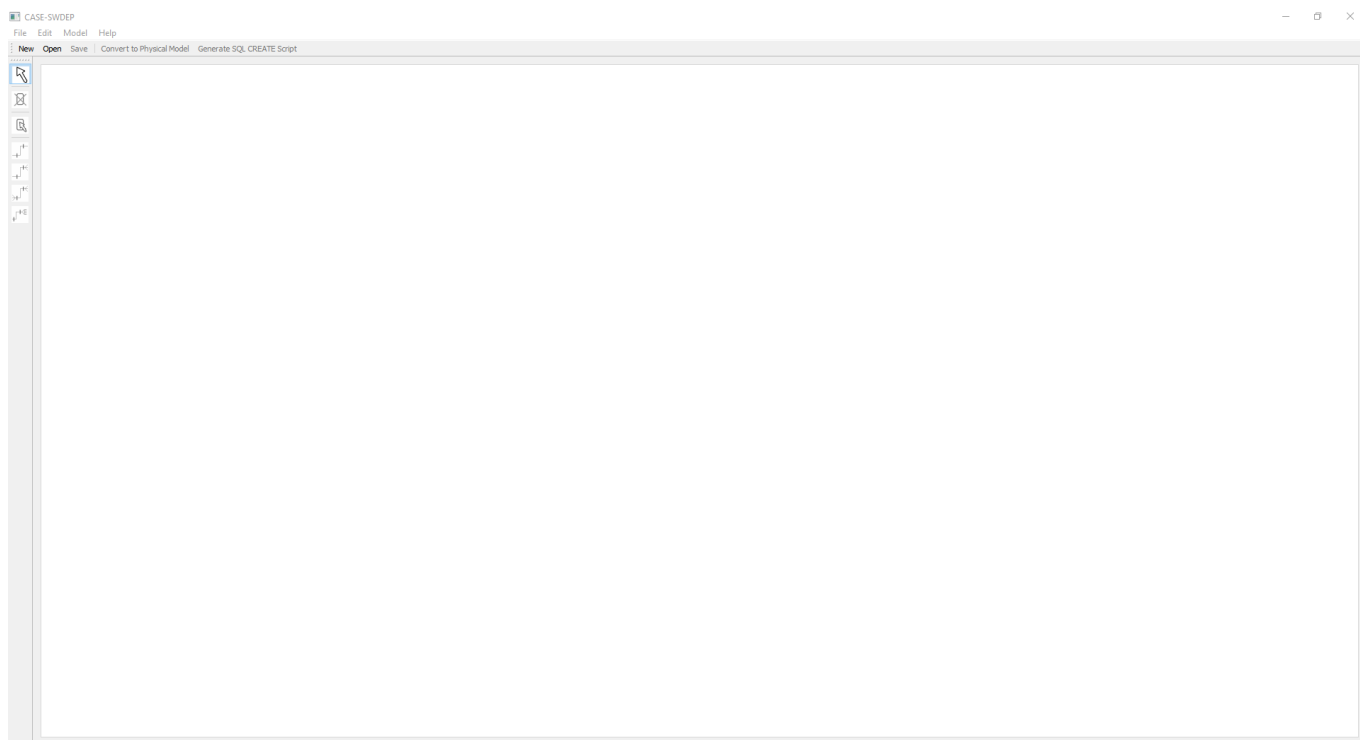
```

После проведённых действий в папке, указанной в файле проекта в поле DLLDESTDIR и DESTDIR, будет создана динамическая библиотека с именем проекта и расширением “.DLL” для ОС семейства Windows и “.SO” для семейства Linux.

Глава 2 Создание простейшего SQL CREATE запроса

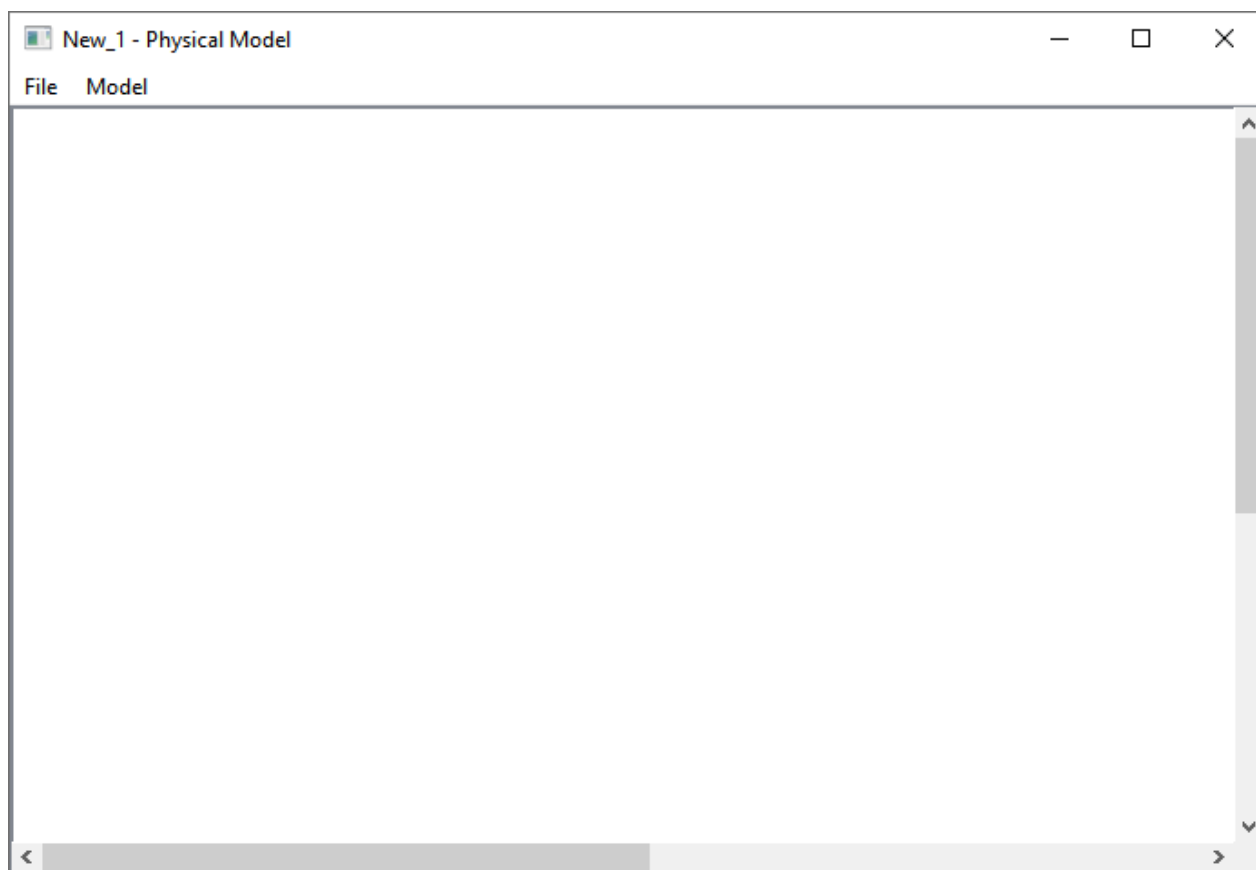
2.1 Пользовательский интерфейс

Главное окно приложения



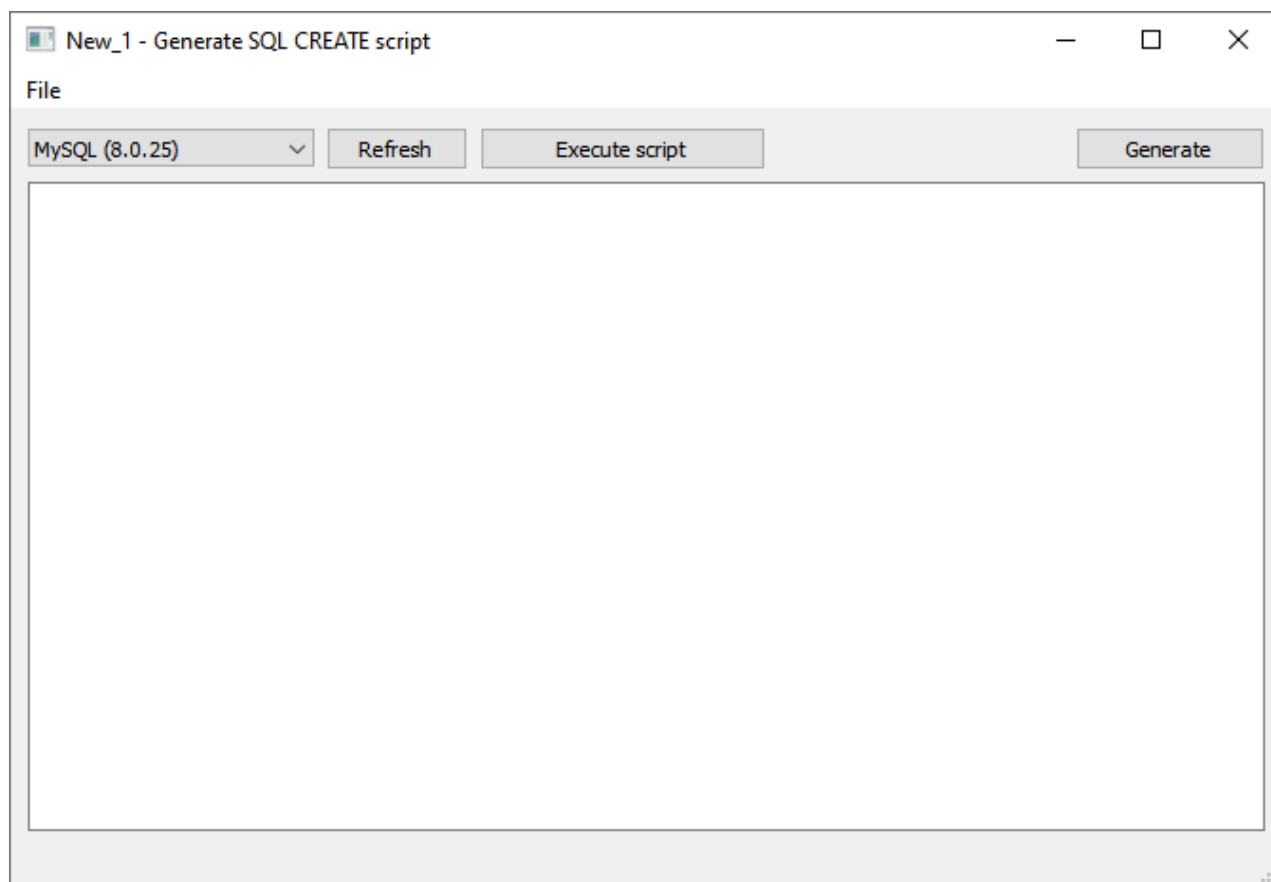
При запуске программы откроется главное окно, на котором располагается несколько панелей инструментов. Некоторые из функций будут недоступны для пользователя, так как сначала необходимо создать новую рабочую область.

Окно физической модели



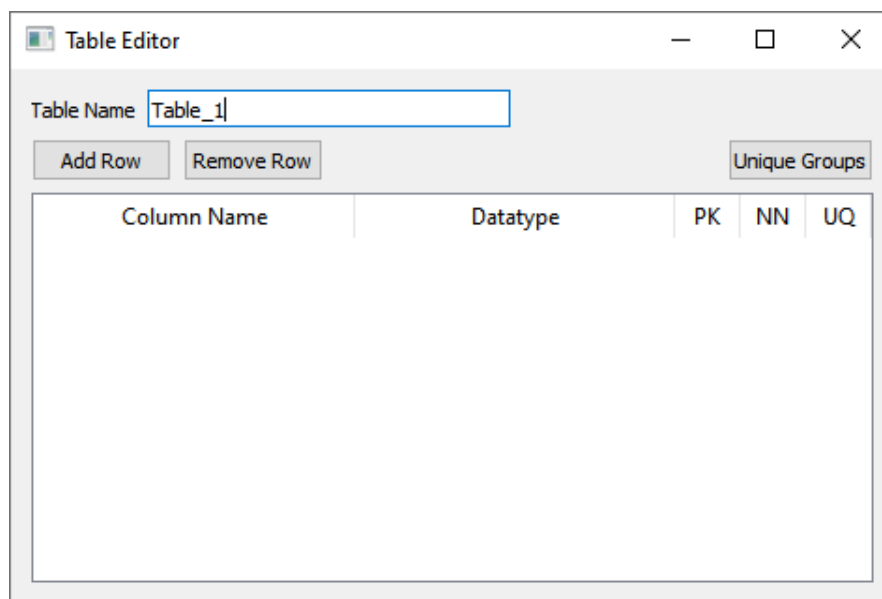
Окно, которое отображает физическую модель данных.

Окно создания SQL запроса



Окно, в котором можно выбрать плагин, на основе которого будет сформирован скрипт, а также подключится к БД и выполнить созданный скрипт.

Окно редактирования таблицы

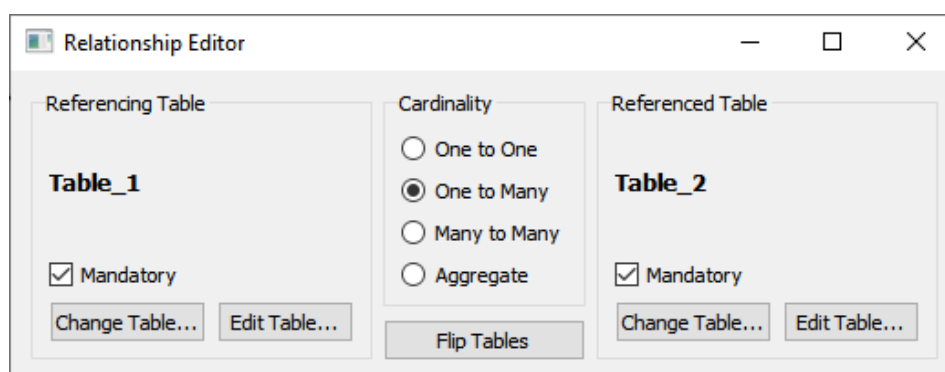


The screenshot shows a window titled "Table Editor" with standard window controls (minimize, maximize, close). Inside the window, there is a text field labeled "Table Name" containing the text "Table_1". Below this, there are two buttons: "Add Row" and "Remove Row". To the right of these buttons is a button labeled "Unique Groups". Below these controls is a table with five columns: "Column Name", "Datatype", "PK", "NN", and "UQ". The table is currently empty.

Column Name	Datatype	PK	NN	UQ
-------------	----------	----	----	----

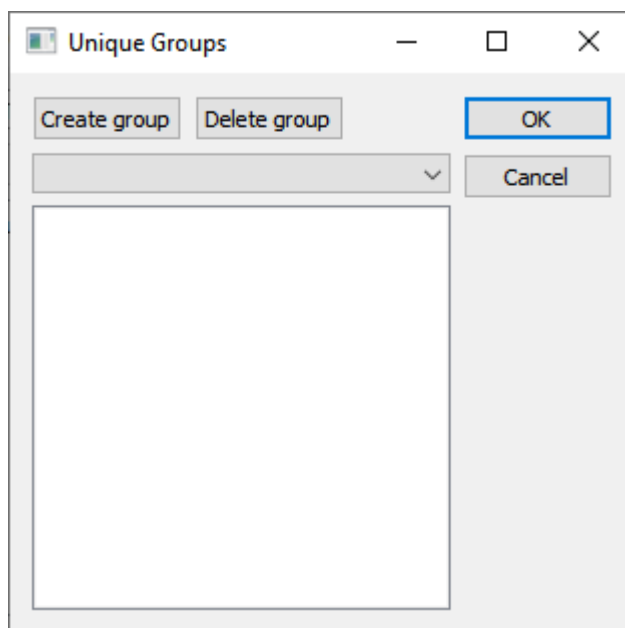
Отображает всю необходимую информацию о таблице и позволяет её редактировать.

Окно редактирования связи

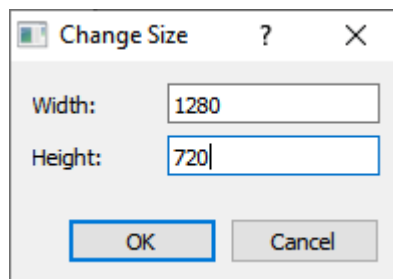


The screenshot shows a window titled "Relationship Editor" with standard window controls. The window is divided into three main sections. On the left, under the heading "Referencing Table", there is a text field containing "Table_1". Below this, there is a checkbox labeled "Mandatory" which is checked. At the bottom of this section are two buttons: "Change Table..." and "Edit Table...". In the center, under the heading "Cardinality", there are four radio button options: "One to One", "One to Many" (which is selected), "Many to Many", and "Aggregate". Below these options is a button labeled "Flip Tables". On the right, under the heading "Referenced Table", there is a text field containing "Table_2". Below this, there is a checkbox labeled "Mandatory" which is checked. At the bottom of this section are two buttons: "Change Table..." and "Edit Table...".

Окно редактирования уникальных групп




Окно изменения размера рабочей области



2.2 Создание концептуальной модели базы данных


В верхней части экрана на панели управления нажать пункт New. Как только рабочая область отобразится, панель инструментов в левой части экрана и некоторые функции панели управления станут активными.

Выбрать опцию создания таблицы можно с помощью:

- кнопки на панели инструментов ;
- Edit – Create Table;
- Клавиши T.

Затем кликаем в любую часть рабочей области и таблица будет создана.

Функция удаления таблицы включается:

- Кнопкой на панели инструментов ;
- Клавишей D.

Затем выбираем таблицу, которую хотим удалить.

Изменение данных в таблице происходит при помощи двойного нажатия ЛКМ по нужной таблице. Откроется окно редактирования таблицы, в котором можно добавлять, удалять и изменять строки, а также создавать уникальные группы.

Создание связей между таблицами

На панели инструментов или в пункте меню Edit – Create Relationship выбираем необходимую связь, далее первой выбираем главную таблицу, а второй – зависимую. Также вид связи можно выбрать с помощью цифр от 1 до 4.

Изменение связей происходит при помощи двойного нажатия ЛКМ по нужной связи. Откроется окно редактирования связи, в котором можно изменить вид связи, поменять направление зависимости и изменить таблицы, между которыми происходит связь.

2.3 Создание физической модели базы данных

После создания концептуальной модели на верхней панели управления выбираем пункт Convert to Physical Model (или Model - Convert to Physical Model...), откроется новое окно, в котором будет отображена физическая модель.

2.4 Создание SQL CREATE запроса

После создания концептуальной модели на верхней панели управления выбираем пункт Generate SQL CREATE Script (или Model - Generate SQL CREATE Script...).

В окне генерации SQL скрипта выбираем плагин под нужную нам СУБД и нажимаем Generate.

2.5 Подключение к БД и выполнение скрипта из программы

До того, как пытаться подключиться, необходимо проверить:

- Наличие на компьютере сервера для определённой СУБД;
- Наличие БД;
- Правильно ли в проекте указаны пути к папкам bin и lib для вашей СУБД.

В окне генерации SQL после создания скрипта нажимаем на кнопку Execute script. В окне авторизации вводим нужные для подключения данные и жмём Подключиться. В случае успеха выведется сообщение о подключении.

2.6 Структура концептуальной модели базы данных в формате CDMOD

[TODO]

2.7 Сохранение файла проекта

Сохранить файл можно:

- С помощью комбинации клавиш Ctrl+S или Ctrl+Shift+S (Сохранить файл как...);
- С помощью пункта панели инструментов Save;
- Выбрав File – Save или File – Save as....

В окне сохранения файла в поле File name пишем имя перед точкой и нажимаем Save.

2.8 Открытие файла проекта

Открыть файл можно:

- Выбрав на панели управления Open;
- Нажав комбинацию клавиш Ctrl+O;
- Выбрав File – Open....

В окне открытия файла выбираем нужный файл с расширением cdmод и кнопку Open.

2.9 Изменение размера рабочей области

На панели инструментов выбираем пункт Model – Change Canvas Size..., в открывшемся окне вводим значения высоты и длины активной рабочей области.