|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ | | | | | | | | |
|  | | |  | | |  | | |
| ФГБОУ ВО «Пермский государственный  национальный исследовательский университет» | | | | | | | | |
|  | | |  | | |  | | |
|  | | ОТЧЕТ  по лабораторной работе № 7 «ORM» по дисциплине «Учебная практика по БД и СУБД» | | | | |  | |
|  | | |  | | |  | | |
|  | Работу выполнил  студент гр. \_\_\_\_\_\_\_\_\_  Зимин И. В.\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2018 | | |  | Проверил  \_\_\_\_\_\_ кафедры МОВС  (доц., ст. преп., асс.)  Постаногов И. С.\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_ 2018 | | |  |
|  |  | | |  |  | | |  |
| Пермь 2018 | | | | | | | | |

1 Постановка задачи

Написать программу, которая на графическом интерфейсе пользователя позволяет просматривать, добавлять, удалять, изменять записи в двух других связанных таблицах из спроектированной ранее базы данных предметной области, без необходимости задания пользователем и отображения ему искусственных ключей. На уровне программного кода CRUD-операции должны выполняться с использованием некоторой ORM-библиотеки.

Провести тестирование корректности работы программы.

2 Проектирование БД

Для решения поставленной задачи требовалось воспользоваться уже ранее спроектированной БД «Кубки мира». В прошлый раз были рассмотрены две не связанные между собой сущности.

Пришло время предоставить возможность просмотра/редактирования двух других, на этот раз связанных сущностей. Это «Гонка» и «Этапы кубка мира». Все 4 сущности представлены на рисунке 2.1.

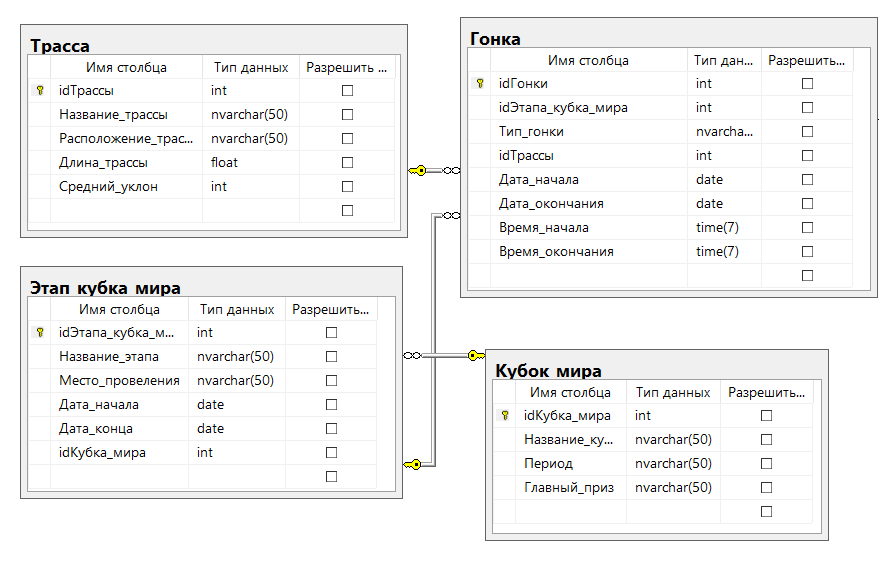


Рисунок 2.1 – БД из 4-х сущностей

«Этапы кубка мира» и «Гонка» находятся в отношении «1:М», то есть в течении одного кубка может быть проведено несколько гонок.

3 Общее описание

Программа представляет из себя Windows Form приложение, реализованное на языке C# в среде VS 2015.

На форме расположены две «сетки» с данными, которые изначально заполняются из БД. На первой «сетке» отображаются данные о гонках, на второй данные о этапах кубка мира. Интерфейс готового приложения представлен на рисунке 3.1.

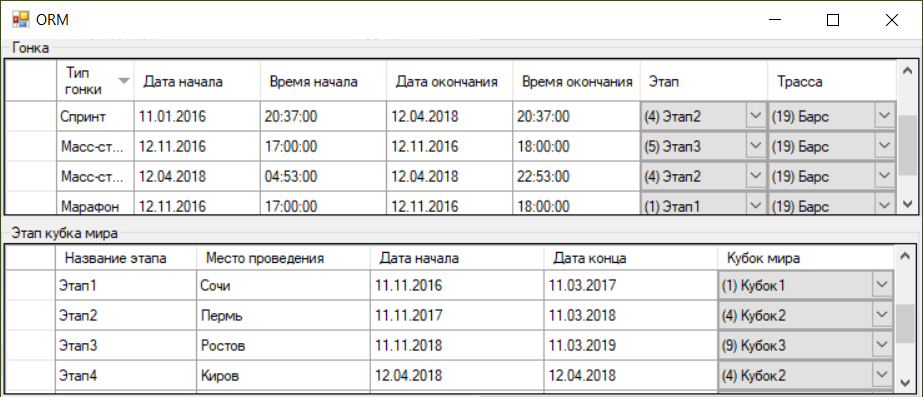


Рисунок 3.1 – Интерфейс приложения

Данное прложение позволяет:

1. Просматривать данные о сущностях
2. Изменять данные о сущностях
3. Добавлять новые сущности
4. Удалять сущности

4 Описание алгоритма

Имеем БД «Кубки мира».

1. Добавляем в наши представления данных на форме, столбцы
   1. Для данных о гонках

* Id Гонки (исскуственный ключ, отображать его не будем)
* Тип гонки
* Дата начала
* Дата окончания
* Время начала
* Время окончания
* Этап
* Трасса
  1. Для данных об этапах
* Id Этапа (исскуственный ключ, отображать его не будем)
* Название этапа
* Место проведения
* Дата начала
* Дата конца
* Кубок мира

1. Берем данные из существующей БД и заносим их в соответствующие представленя на форме
2. Обрабатываем переходы с ячейки на ячейку в пределах одной строки в представлениях на форме.
   1. Прешли с ячейки данные в которой были изменены: размещаем в той ячейке красненький круглешок с восклицательным знаком и подписью «Данные изменены но не сохранены».
   2. Перешли с ячейки не меняя в ней значения: ничего не предпринимаем.
3. Обрабатываем переход со строки на строку в пердставлениях на форме
   1. Перешли со строки, которая была последняя (еще не существующаяя в БД): добавляем внесенные на представление данные в БД. Если данные были внесены таким образом, что сформировалсась строка эдентичная какой либо другой строке в БД. То устанавливаем запрет на добавление.
   2. Перешли с ранее существующей строки, данные в которой были изменены: изменяем соответсвующие данные в БД. Если данные были изменены таким образом, что сформировалсась строка, эдентичная какой-либо другой строке в БД. То устанавливаем запрет на изменение.
   3. Если перешли со строки в которой ничего не было изменено: ничего не предпринимаем.
4. Обрабатываем удаление какой-либо строки. Реакция на нажатие на клавиатуре клавиши Delete.
   1. Попытка удалить последнюю (еще не существующую БД) строку: ничего не предпринимаем.
   2. Попытка удалить любую другую строку: удаляем ее из БД.
5. Обрабатываем коорректный ввод данных а именно:

* Пустые поля при вводе данных не допустимы
* Ввод данных не соотвутствующих установленному типу не допустим

5 Особенности реализации на C#

Для решения задачи необходимо было воспользоваться некоторой ORM-библиотекой. Поэтому, для начала введем понятие ORM:

объектно-реляционное отображение (ORM) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

В разработанной программе была использована ORM-библиотека Entity Framework 6.2., установить которую можно на вкладке «Обзор» в управлении пакетами NuGet текущего проекта.

Для разработки приложения была использована Модель ADO.NET EDM

С помощью средств модели EDM можно создать концептуальную модель на основе существующей базы данных, а затем создать ее графическое представление и внести нужные изменения. Либо можно сначала создать концептуальную модель с помощью графических средств, а затем создать базу данных, которая поддерживает эту модель.

Для работы с данной моделью необходимо ее добавить в проект.

Для этого из контекстного меню проекта выбрать пункт «Добавить новый элемент». Затем перейти на вкладку «Данные» и выбрать в списке предложенных Модель ADO.NET EDM. После чего требуется указать содержимое модели (в моем случае выбор пал на элемент «Code First из базы данных», который создает модель в конструкторе EF на основе существующей базы данных).

Затем необходимо создать подключение, указав имя сервера и имя БД. Выбрать нужные таблицы (в моем случае это таблицы «Кубок мира», «Этап кубка мира», «Гонка» и «Трассы»

Далее начинается этап программирования. После создания формочки, на которой расположены два элемента DataGridView для таблиц «Гонки» и «Этапы кубка мира» соответственно, необходимо заполнить их данными из БД.

Для начала необходима очистка коллекций строк и столбцов (в частности, поможет при каскадном обновлении отдельных полей):

dgvHeader.Rows.Clear();

dgvHeader.Columns.Clear();

Для добавления столбцов на элементы DataGridView следует воспользоваться методом Add() коллекции Columns. Который имеет несколько перегрузок.

Если требуется добавить текстовый столбец, который необходимо скрыть (то есть сделать так, чтобы он не отображался на DataGridView) следует передать в конструктор экземпляр класса DataGridViewTextBoxColumn, где в свойство Name следует передать имя столбца, по которому в дальнейшем будет происходить обращение к нему. И свойство Visible установить в false, например:

dgvStage.Columns.Add(new DataGridViewTextBoxColumn

{

Name = "idЭтапа",

Visible = false

});

Если требуется создать столбец, в котором будут содержаться только текстовые значения, следует воспользоваться следующей перегрузкой метода Add()

dgvStage.Columns.Add("Название\_этапа", "Название этапа");

Где первый параметр передается свойству Name, а второй будет являться заголовком самого столбца в DataGridView (свойство HeaderText)

Столбец DataGridView, может содержать не только текст, но и различные элементы, например ComboBox, DateTimePicker и другие. Для этого необходимо, при создании объекта в качестве параметра метода Add() передать объект нужного класса элемента, при этом, объект должен быть инициализирован.

Для инициализации создадим новый объект класса WorldCupsContext, который автоматически создался при добавлении модели данных в проект.

Доступ к нужной таблице из БД можно взять следующим образом: <объект WorldCupsContext >.<имя таблицы>. Таблица представлена как коллекция.

Доступ к конкретным полям определенной строки таблицы из БД, можно также взять следующим образом: <элемент коллекции>.<название поля>.

DataGridView имеет коллекцию всех строк Rows, которая в свою очередь содержит метод Add() для добавления новой строки. В качестве параметров указываются конкретные значения полей строки, в том порядке, в каком были ранее созданы столбцы.

Обратившись к коллекции Rows по индексу (номер строки), можно также обратиться к коллекции полей по имени столбца, которая в свою очередь имеет свойство Value. Таким образом можно установить значение только для одного поля конкретной строки, например:

dgvRace.Rows[i].Cells["Тип\_гонки"].Value = “Спринт”

Поля в DataGridView с элементами ComboBox заполняются следующим образом:

var listCups = ctx.Кубок\_мира.ToDictionary(item => item.idКубка\_мира, item => "(" + item.idКубка\_мира + ") " + item.Название\_кубка);

cups.DataSource = listCups.Values.ToList();

Для добавления удаления и изменения строк понадобились следующие события DataGridView: RowValidaing, которое срабатывает при переключении на другую ячейку DataGridView. Чтобы такого не было, используется проверка IsCurrentRowDirty, которое вернет true, если в строке нет незафиксированных изменений.

Проверка на корректность значений выполняется с помощью перебора значений столбцов в добавляемой / обновляемой строке. Для установки сообщения об ошибке используется свойство ErrorText строки DataGridView, при этом значений свойства CancelEventsArgs.Cancel устанавливает в true (т.е. отмена события).

Если же CancelEventsArgs.Cancel имеет значение false (т.е. данные корректны), то выполняется добавление / обновление строки.

Считываем значение первичного ключа (который автоматически инкрементируется в БД). Если оно не null, значит выполняется операция обновления, иначе – добавление.

var idStage = (int?)row.Cells["idЭтапа"].Value;

if (idStage.HasValue)

{//Обновляем}

else {//Добавляем}

**Добавление:**

Для того чтобы добавить новые данные в БД создаем объект WorldCupsContext. Создаем новый объект таблицы Гонка и инициализируем поля в конструкторе. При считывании значений полей из DataGridView нужно делать явное приведение типов.

Для добавлении записи коллекция таблицы имеет метод Add(), принимающий объект таблицы.

Для сохранения изменений вызывается метод SaveChanges() класса WorldCupsContext.

В тег добавленной строки помещаем созданный объект таблицы.

var race = new Гонка

{

Тип\_гонки = typeTrack,

Дата\_окончания = dateEnd,

Дата\_начала = dateBegin,

Время\_начала = bTime,

Время\_окончания = eTime,

idЭтапа\_кубка\_мира = idStage,

idТрассы = idTrack

};

ctx.Гонка.Add(race);

row.Tag = race;

row.Cells["idГонки"].Value = race.idГонки;

ctx.SaveChanges();

**Обновление:**

Создаем объект WorldCupsContext. Считывается объект таблицы Гонка из тега строки DataGridView, при этом выполняется явное приведение к типу Гонка.

Считанный элемент прикрепляется к текущему контексту:

ctx.Гонка.Attach(race);

Устанавливаются новые значения для полей считанного объекта таблицы из тега.

Далее измененный объект перезаписывается в тег строки DataGridView.

Для сохранения изменений вызывается метод SaveChanges() класса WorldCupsContext.

var cupId = (int?)row.Cells["idГонки"].Value;

if (cupId.HasValue)

{

ctx.Гонка.Attach(r);

r.Тип\_гонки = typeTrack;

r.Дата\_окончания = dateEnd;

r.Время\_начала = bTime;

r.Время\_окончания = eTime;

r.Дата\_начала = dateBegin;

r.idЭтапа\_кубка\_мира = idStage;

r.idТрассы = idTrack;

row.Tag = r;

row.Cells["idГонки"].Value = r.idГонки;

}

ctx.SaveChanges();

**Удаление:**

Используется событие DataGridView UserDeletingRow, которое срабатывает, при нажатии клавиши Del. Сначала выполнится код, привязанный к событию, потом удалится сама строка.

Создаем объект WorldCupsContext. Считывается объект таблицы Гонка из тега строки DataGridView, при этом выполняется явное приведение к типу Гонка.

Считанный элемент прикрепляется к текущему контексту:

ctx.Гонка.Attach(r);

Удаление записи из коллекции таблицы выполняется с помощью метода Remove, в качестве аргумента передается объект таблицы:  
ctx.Гонка.Remove(r);

Для сохранения изменений вызывается метод SaveChanges() класса WorldCupsContext.

var cupId = (int?)e.Row.Cells["idГонки"].Value;

if (!cupId.HasValue) return;

using (var ctx = new WorldCupsContext())

{

var r = (Гонка) e.Row.Tag;

ctx.Гонка.Attach(r);

ctx.Гонка.Remove(r);

ctx.SaveChanges();

}

После добавления, удаления, изменения строк в таблице Этапы\_кубка\_мира

заново инициализируем DataGridView относящийся к таблице «Гонки»

6 Тестирование программы

* Для тестирования потребуется:
* Созданная БД
* Минимально наполненные связанные таблицы «Гонка» и «Этап кубка мира», а также таблицы «Кубок мира» и «Трассы».
* Разработанное приложение для просмотра/изменения данных в этих таблицах

Наполнение БД

Таблицы, которые невозможно изменить из приложения:

Таблица 6.1 – Кубки мира.

|  |  |  |  |
| --- | --- | --- | --- |
| idКубка\_мира | Название\_кубка | Период | Главный\_приз |
| 1 | Кубок1 | 16/17 | БМВ |
| 2 | Кубок2 | 17/18 | БХГ |

Таблица 6.2 – Трассы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| idТрассы | Название\_трассы | Расположение\_трассы | Длина\_трассы | Средний\_уклон |
| 1 | Трасса1 | Р1 | 12 | 10 |
| 2 | Трасса2 | Р2 | 15 | 8 |

Таблица 6.3 – Гонка

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| idГонки | Тип\_гонки | Дата\_начала | Время\_начала | Дата\_окончания | Время\_окончания | idЭтапа\_кубка\_мира | idТрассы |
| 1 | Спринт | 12-12-2016 | 10:10:00 | 12-12-2016 | 12:00:00 | 1 | 1 |

Таблица 6.4 – Этапы кубка мира

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| idЭтапа\_кубка\_мира | Название\_этапа | Место\_проведения | Дата\_начала | Дата\_конца | idКубка\_мира |
| 1 | Этап1 | Сочи | 12-11-2016 | 12-01-2017 | 1 |

**Тест№1.** Заполнение из БД.

* Суть теста: проверить, что данные из БД правильно отображаются в представления на форме
* Последовательность шагов тестировании:
  + Запустить приложение
  + Сравнить данные в БД и данные на сетках формы.
* Входные данные: Таблицы 6.2-6.3
* Ожидаемый результат: Все считано верно
* Фактический результат: совпал с ожидаемым.

**Тест№2.** Добавление не существующей строки.

* Суть теста: проверить, что данные внесенные в таблицу на форме добавляются в БД.
* Последовательность шагов тестировании:
  + Запустить приложение
  + Внести новые данные в последнюю строку
  + Перейти на любую другую строку
  + Проверить в БД, что данные добавлены
* Входные данные: в таблицу «Гонка» внесенные данные

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Масс-старт | 13-12-2016 | 10:10:00 | 12-12-2016 | 12:00:00 | 1 | 2 |

* Ожидаемый результат: Строка успешно добавлена в БД
* Фактический результат: совпал с ожидаемым.

**Тест3.** Добавление существующей строки.

* Суть теста: проверить, что при внесении в новую строку данных, полностью соответствующих какой-либо строке в БД, будет ошибка с комментарием «Идентичная строка уже есть в БД»
* Последовательность шагов тестировании:
  + Запустить приложение
  + Внести данные, соответствующие какой-либо строке в последнюю строку
  + Перейти на любую другую строку
  + Проверить в БД, что данная строка не добавляется
* Входные данные: в таблицу «Гонка» внесенные данные

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Масс-старт | 13-12-2016 | 10:10:00 | 12-12-2016 | 12:00:00 | 1 | 2 |

* Ожидаемый результат: ошибка, добавить нельзя
* Фактический результат: совпал с ожидаемым.

**Тест№4.** Изменение данных.

* Суть теста: проверить, что данные измененные в сетке на форме изменяются и в БД.
* Последовательность шагов тестировании:
  + Запустить приложение
  + Изменить какое либо поле (или все) в любой существующей строке
  + Перейти на любую другую строку
  + Проверить в БД, что сделаны соответствующие изменения
* Входные данные: в таблице «Гонка» изменим «Масс-старт» на «Спринт»
* Ожидаемый результат:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Спринт | 13-12-2016 | 10:10:00 | 12-12-2016 | 12:00:00 | 1 | 2 |

* Фактический результат: совпал с ожидаемым.

**Тест№5.** Каскадное изменение.

* Суть теста: проверить, что при изменении данных таблицы, которые присутствуют и в зависимой таблице меняются везде.
* Последовательность шагов тестирования:
  + Запустить приложение
  + Изменить название этапа в таблице «Этапы кубка мира» на «Этап11»
  + Перейти на любую другую строку
  + Проверить, что в таблице «Гонка» были сделаны соответствующие изменения.
  + Входные данные: в таблице «Этапы кубка мира» изменим название с «Этап1» на «Этап11»
* Ожидаемый результат:

Таблица «Гонка»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Спринт | 13-12-2016 | 10:10:00 | 12-12-2016 | 12:00:00 | Трасса1 | Этап11 |

Таблица «Этапы кубка мира»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| idЭтапа\_кубка\_мира | Название\_этапа | Место\_проведения | Дата\_начала | Дата\_конца | idКубка\_мира |
| 1 | Этап11 | Сочи | 12-11-2016 | 12-01-2017 | Кубок1 |

* Фактический результат: совпал с ожидаемым.

**Тест№6.** Удаление строки без связей.

* Суть теста: проверить, удалив строку не связанную ни с какими другими, она удаляется и в БД.
* Последовательность шагов тестировании:
  + Запустить приложение
  + Удалить нужную строку
  + Проверить соответствующие изменения в БД
* Входные данные: удалим вторую строку из таблицы «Гонка» (была добавлена во 2-м тесте)
* Ожидаемый результат: Строка успешно удалена из БД
* Фактический результат: совпал с ожидаемым.

**Тест7.** Удаление связанной строки.

* Суть теста: проверить, что при удалении строки, которая используется в других таблицах выдается ошибка с комментарием: «используется в БД»
* Последовательность шагов тестировании:
  + Запустить приложение
  + Удалить нужную строку
  + Проверить что появилась ошибка и удаление прервано.
* Входные данные: попытаемся удалить первую строку из таблицы «Этап кубка мира» (используется в таблице «Гонка» в первой строке)
* Ожидаемый результат: ошибка, удалить нельзя
* Фактический результат: совпал с ожидаемым.