|  |  |  |
| --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ | | |
|  |  | |
| ФГБОУ ВО «Пермский государственный  национальный исследовательский университет» | | |
|  |  | |
| ОТЧЕТ  по лабораторной работе «Объектно-реляционное отображение» | | |
|  |  | |
| Работу выполнил  студент гр. \_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_кафедры МОВС  (доц., ст.преп., асс.)  ФИО \_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2018 |
| Минин А.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 данной задачи будет использоваться ранее спроектированная нормализованная база данных «Автосервис». Схема базы данных представлена на рисунке 2.1.

База данных «Автосервис» включает в себя 10 таблиц:

1. Шапка наряд заказ
2. Сотрудник
3. Заказчик
4. Автомобиль
5. Информация об автомобиле
6. Модель
7. Служба качества
8. Оценка
9. Работ
10. Стоимость

В данном случае, для реализации операций чтения, добавления, изменения и удаления будут использоваться таблицы «Автомобиль» и «Шапка наряд-заказ», которые имеют связь 1:М.

Структуры таблиц «Автомобиль» и «Шапка наряд-заказ» представлены в таблицах 2.1 и 2.2 соответственно.

| Имя столбца | Тип данных | Разрешить NULL |
| --- | --- | --- |
| ID\* | int | Нет |
| ID инфо | int | Нет |
| Гос. номер | nvarchar (9) | Нет |

Таблица 2.1 – Автомобиль.

| Имя столбца | Тип данных | Разрешить NULL |
| --- | --- | --- |
| № наряд заказа\* | int | Нет |
| Дата обращения | date | Нет |
| Пробег | int | Нет |
| ID сотрудника | int | Нет |
| ID заказчика | int | Нет |
| ID автомобиля | int | Нет |

Таблица 2.2 – Шапка наряд-заказ.

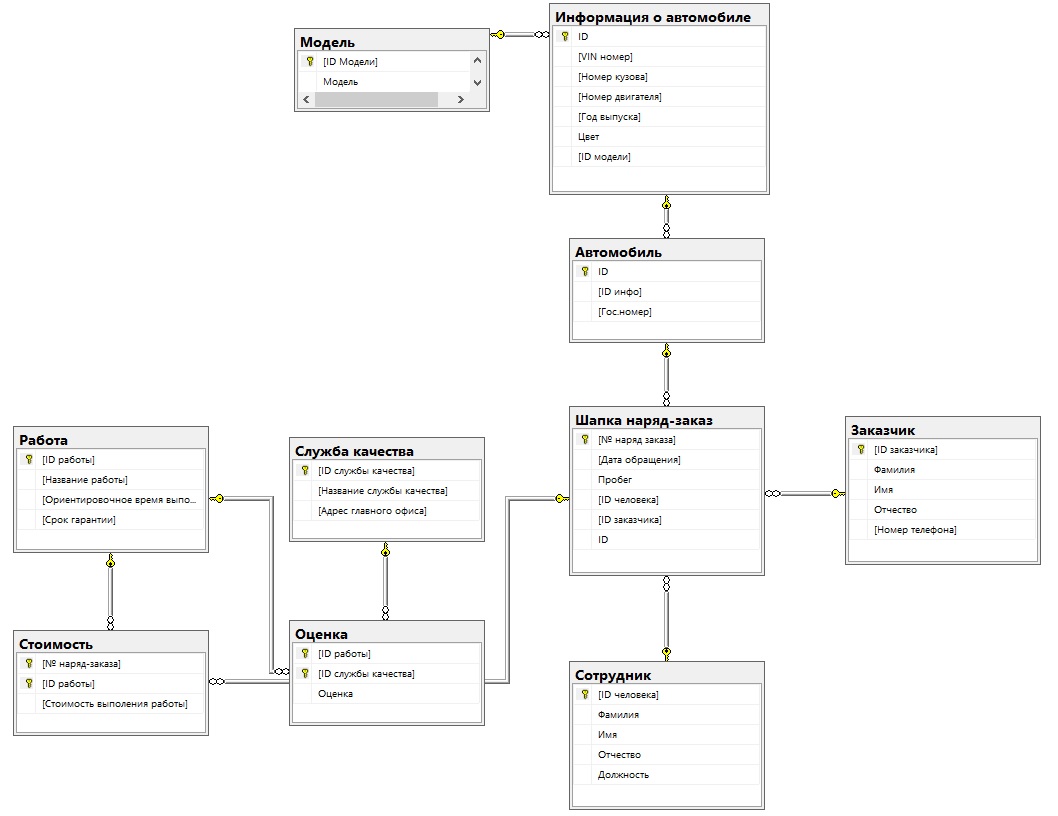


Рисунок 2.1 – Схема БД

3 Общее описание

Программа будет представлять Windows Form приложение, реализованное на языке C#. Программа будет иметь форму для работы с данными.

При запуске приложения показывается форма с двумя контейнерами в виде таблиц, которые будут заполнены данными из соответствующих таблиц базы данных.

С любой из таблиц пользователь сможет выполнять операции по удалению элементов, добавлению новых элементов и изменению существующих.

При вводе некорректных данных, при незаполненных полях будет выводиться сообщение об ошибке.

Вид готового приложения показан на рисунке 3.1.

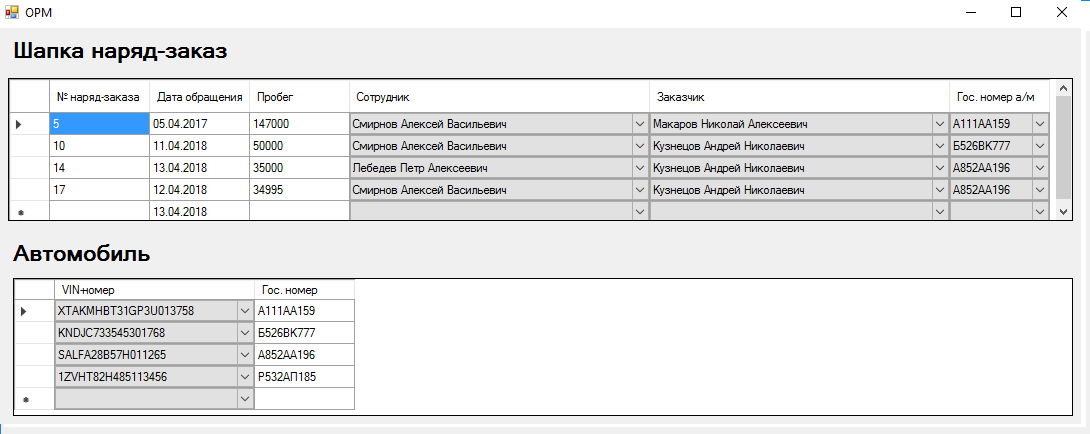


Рисунок 3.1 – Вид приложения

4 Описание алгоритма

Рассматриваем БД «Автосервис».

1. Добавляем в табличные представления данных на форме, столбцы.
   1. Шапка наряд-заказ

* № наряд заказа (первичный ключ, отображается)
* Дата обращения
* Пробег
* Сотрудник
* Заказчик
* Гос. номер
  1. Автомобиль
* ID а/м (исскуственный ключ, не отображается)
* VIN-номер
* Гос. номер

1. Берем данные из существующей БД и заносим их в соответствующие представления на форме.
2. Обрабатываем переход со строки на строку в представлениях на форме
   1. Перешли с последней строки (еще не существующая в БД): добавляем внесенные на представление данные в БД.
   2. Перешли с ранее существующей строки, данные в которой были изменены: изменяем соответсвующие данные в БД.
3. Обрабатываем удаление какой-либо строки. Реакция на нажатие клавиши Del. При попытке удалить любую строку, удаляем соответствующие данные и из БД.
4. Обрабатываем корректный ввод данных:

* Пустые поля при вводе данных недопустимы. Выводим сообщение с текстом ошибки и указанием имени столбца, в котором допущена ошибка.
* Ввод данных не соответствующих установленному типу также недопустим. Выводим сообщение с текстом ошибки и указанием имени столбца, в котором допущена ошибка.

1. Связанная таблица.

Если в таблицу «Автомобили» был добавлен новый автомобиль, то тогда в таблице «Шапка наряд-заказ» в столбце «Гос. номер» в выпадающем списке появится новый гос. номер. Если же в таблице «Автомобили» гос. номер был просто обновлен, то в таблице «Шапка наряд-заказ» в столбце «Гос. номер» произойдет изменение гос. номера и в выпадающем списке, и у всех строк с данным выбранным гос. номером.

5 Особенности реализации на C#

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

При разработке программы будет использоваться ORM-библиотека Entity Framework.

1. Подключение библиотеки.

Библиотеку необходимо подключить к текущему проекту Visual Studio, предварительно установив ее с помощью NuGet (система управления пакетами для платформ разработки Microsoft).

Также необходимо для проекта создать элемент «Модель ADO.NET EDM» (Добавить элемент -> Элементы Visual C# -> Данные). В данном случае данный файл имеет имя OpendataContext.

Далее, в открывшемся окне, нужно выбрать содержимое модели – «Code First из базы данных». «Code First из базы данных» - создает модель в конструкторе EF на основе существующей базы данных.

Следующим шагом создается подключение к БД, указывается, что модель должна содержать таблицы.

На этом необходимые действия по настройке библиотеки закончены.

1. Представление данных

Данные будут храниться в контейнере DataGridView (настраиваемая сетка).

Предварительная очистка коллекций строк и столбцов:

dgvHeader.Rows.Clear();

dgvHeader.Columns.Clear();

Добавить столбец в DataGridView можно вызвав метод Add() коллекции Columns.

Столбец можно добавить, указав в качестве параметров метода Add() название столбца и отображаемый текст:

dgvHeader.Columns.Add("Mileage", "Пробег");

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

dgvHeader.Columns.Add(new DataGridViewComboBoxColumn

{

Name = "Avto",

HeaderText = "Гос. номер а/м",

DataSource = lstAvto

});

1. Считывание данных

Создаем новый объект класса OpendataContext (тот, что добавляли в самом начали).

Доступ к нужной таблице БД можно получить через свойство объекта OpendataContext: <объект OpendataContext >.<имя таблицы>. Таблица представлена как коллекция.

Доступ к конкретным полям определенной строки таблицы из БД, можно также получить через свойство: <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:

dgvHeader.Rows[i].Cells["Worker"].Value = <Значение>

Просматривая коллекцию таблицы с помощью цикла foreach, заполняем строки DataGridView, а в тег строки сохраняем элемент коллекции таблицы.

1. Получение данных для столбца с ComboBox

Данные полей сотрудник, заказчик, гос. номер а/м, VIN-номер берутся из других таблиц. Они хранятся в словарях по типу <ID записи в БД> <Отображаемый текст>. За заполнение словарей отвечают функции InitializeCustomer(), InitializeAvto(), InitializeWorker(), InitializeInfo().

При заполнении значения поля столбца DataGridView c ComboBox значения из словарей выносятся в отдельный список:

List<string> lstWorker = new List<string>();

lstWorker = dictWorker.Values.ToList();

Само заполнение значения:

dgvHeader.Rows[i].Cells["Worker"].Value = (dgvHeader.Rows[i].Cells["Worker"] as DataGridViewComboBoxCell).Items[FindPosition(dictWorker, (int)item.ID\_человека)];

Функция FindPosition(<словарь>, <ID элемента в БД>) возвращает порядковую позицию элемента в словаре, у которого значение ключа = ID элемента из БД.

1. Добавление и обновление записи

Используется событие 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 num = (int?)row.Cells["Num"].Value;

if (num.HasValue)

{//UPDATE}

else {//CREATE}

* 1. Добавление:

Создаем объект OpendataContext. Создаем новый объект таблицы Шапка\_наряд\_заказ 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 OpendataContext.

В тег добавленной строки помещаем созданный объект таблицы.

В ячейку с номером наряд-заказа прописывается идентификатор только что добавленной строки из БД.

Аналогично для таблицы «Автомобиль».

using (var ctx = new OpendataContext())

{

var record = new Шапка\_наряд\_заказ

{

Дата\_обращения = (DateTime)row.Cells["Date"].Value,

Пробег = Convert.ToInt32(row.Cells["Mileage"].Value),

ID\_человека = GetKeyByValue(dictWorker, (string)(row.Cells["Worker"] as DataGridViewComboBoxCell).FormattedValue),

ID\_заказчика = GetKeyByValue(dictCustomer, (string)(row.Cells["Customer"] as DataGridViewComboBoxCell).FormattedValue),

ID = GetKeyByValue(dictAvto, (string)(row.Cells["Avto"] as DataGridViewComboBoxCell).FormattedValue)

};

ctx.Шапка\_наряд\_заказ.Add(record);

ctx.SaveChanges();

row.Tag = record;

row.Cells["Num"].Value = (int)record.C\_\_наряд\_заказа;

}

* 1. Обновление

Создаем объект OpendataContext. Считывается объект таблицы Шапка\_наряд\_заказ из тега строки DataGridView, при этом выполняется явное приведение к типу Шапка\_наряд\_заказ.

Считанный элемент прикрепляется к текущему контексту:

ctx.Шапка\_наряд\_заказ.Attach(record);

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 OpendataContext.

using (var ctx = new OpendataContext())

{

var record = (Шапка\_наряд\_заказ)row.Tag;

ctx.Шапка\_наряд\_заказ.Attach(record);

record.Дата\_обращения = (DateTime)row.Cells["Date"].Value;

record.Пробег = Convert.ToInt32(row.Cells["Mileage"].Value);

record.ID\_человека = GetKeyByValue(dictWorker, (string)(row.Cells["Worker"] as DataGridViewComboBoxCell).FormattedValue);

record.ID\_заказчика = GetKeyByValue(dictCustomer, (string)(row.Cells["Customer"] as DataGridViewComboBoxCell).FormattedValue);

record.ID = GetKeyByValue(dictAvto, (string)(row.Cells["Avto"] as DataGridViewComboBoxCell).FormattedValue);

row.Tag = record;

ctx.SaveChanges(); }

Функция GetKeyByValue() принимает в качестве параметров словарь типа <int, string> и строку. Она выполняет в словаре поиск ключа по значению, и возвращает ключ. Это необходимо при обновлении, когда из поля DataGridView с ComboBox считывается установленное значение.

1. Удаление

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

Создаем объект OpendataContext. Считывается объект таблицы Шапка\_наряд\_заказ из тега строки DataGridView, при этом выполняется явное приведение к типу Шапка\_наряд\_заказ.

Считанный элемент прикрепляется к текущему контексту:

ctx.Шапка\_наряд\_заказ.Attach(record);

Удаление записи из коллекции таблицы выполняется с помощью метода Remove, в качестве аргумента передается объект таблицы: ctx.Шапка\_наряд\_заказ.Remove(record);

Для сохранения изменений вызывается метод SaveChanges() класса OpendataContext.

При сохранении изменений, удаление записи из самой БД может завершиться с ошибкой (то что удаляемая запись задействована в других таблицах). Будет сгенерировано исключение System.Data.Entity.Infrastructure.DbUpdateException, которое обрабатывается с помощью блока try {} catch {}.

Если исключение было сгенерировано, то выводим сообщение и отменяем удаление, посредством установки свойства CancelEventsArgs.Cancel в false.

5. Тестирование программы

Рассмотрим следующий набор тестов:

Предварительное наполнение данными таблиц:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № наряд-заказ | Дата обращения | Пробег | ID человека | ID заказчика | ID а/м |
| 1 | 2017-04-05 | 147000 | 1 | 1 | 1 |
| 2 | 2018-04-11 | 50000 | 2 | 2 | 2 |

Таблица 5.1 – Шапка наряд-заказ

|  |  |  |
| --- | --- | --- |
| ID а/м | ID инфо | Гос. номер |
| 1 | 1 | A111AA159 |
| 2 | 2 | Б526ВК777 |
| 3 | 3 | А852АА19 |

Таблица 5.2 - Автомобиль

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID человека | Фамилия | Имя | Отчество | Должность |
| 1 | Смирнов | Алексей | Васильевич | Оператор по ремонту |
| 2 | Лебедев | Петр | Алексеевич | Оператор по ремонту |

Таблица 5.3 - Сотрудник

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID заказчика | Фамилия | Имя | Отчество | Номер телефона |
| 1 | Федоров | Никита | Валентинович | 89091231231 |
| 2 | Кузнецов | Андрей | Николаевич | 89991231231 |

Таблица 5.4 – Заказчик

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | VIN номер | Номер кузова | Номер двигателя | Год выпуска | Цвет | ID модели |
| 1 | XTAKMHBT31GP3U013758 | 9873657 | 2432254 | 2005 | Желтый | 1 |
| 2 | KNDJC733545301768 | 4365718 | 8765433 | 2013 | Голубой | 2 |
| 3 | SALFA28B57H011265 | 8764322 | 1345355 | 2015 | Черный | 3 |
| 4 | 1FALP62W4WH128703 | 1234523 | 1231234 | 2015 | Черный | 1 |

Таблица 5.5 – Информация об автомобиле

|  |  |
| --- | --- |
| ID модели | Название модели |
| 1 | Ford Focus |
| 2 | Kia Spectra |
| 3 | Daewoo Nexia |

Таблица 5.6 - Модели

**Тест 1.** Чтение данных из БД.

Суть теста: проверить, что данные из БД правильно отображаются в представления на форме.

Шаги тестирования:

* Запустить приложение.
* Сравнить данные в БД и данные на сетках формы.

Ожидаемый результат: данные считаны и отображаются верно.

Фактический результат: совпало с ожидаемым результатом.

**Тест 2.** Добавление не существующей строки в таблицу

Суть теста: проверить, что данные внесенные в таблицу на форме добавляются в БД.

Шаги тестирования:

* Запустить приложение.
* Внесем в таблицу 5.1 данные вида:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 2017-01-05 | 34999 | 2 | 1 | 1 |

* Перейти на любую другую строку.
* Проверить, что введенные данные добавлены в БД.

Ожидаемый результат: введенные данные отображаются на форме и добавились в соответствующую таблицу БД.

Фактический результат: совпало с ожидаемым результатом.

**Тест 3.** Обновление строки в таблице

Суть теста: проверить, что новые данные внесенные в таблицу на форме обновляются в БД.

Шаги тестирования:

* Запустить приложение.
* Изменим строку в таблице 5.1 следующим образом:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2017-04-05 | 155000 | 1 | 2 | 1 |

* Перейти на любую другую строку.
* Проверить, что введенные данные обновлены в БД.

Ожидаемый результат: внесенные изменения в строке на форме появятся у записи соответствующей таблицы БД.

Фактический результат: совпало с ожидаемым результатом.

**Тест 4.** Удаление строки

Суть теста: проверить, что строка, у которой данные не связаны с другими таблицами в БД удалится, а при удалении строки, первичный ключ которой задействован в других таблицах, будет выведено сообщение о невозможности удаления такой строки.

Шаги тестирования:

* Запустить приложение.
* Удалить из таблицы 5.1 запись:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 2017-01-05 | 34999 | 2 | 1 | 1 |

* Удалить из таблицы 5.2 запись:

|  |  |  |
| --- | --- | --- |
| 1 | 1 | A111AA159 |

* Проверить, что данные удалились из таблицы БД

Ожидаемый результат: Запись из таблицы 5.1 удалится с формы и из БД. Запись из таблицы 5.2 не удалится, ни с формы, ни из БД, будет выведено сообщение «Удалить невозможно. Данный объект используется в других таблицах БД.»

Фактический результат: совпало с ожидаемым результатом.

**Тест 5.** Добавление новой записи в связанную таблицу.

Суть теста: проверить, что данные в столбце таблицы с выпадающим списком обновятся, при добавлении новой записи в связанную таблицу.

Шаги тестирования:

* Запустить приложение.
* Добавить в таблицу 5.2 запись:

|  |  |  |
| --- | --- | --- |
| 4 | 4 | Н000НН777 |

* Проверить, что в таблицу 5.1 столбце «Гос. Номер» у выпадающего списка появился новый элемент «Н000НН777».
* Проверить, что данные добавились и в БД.

Ожидаемый результат: в таблице 5.1 у столбца «Гос. Номер» с выпадающим списком появится новый элемент «Н000НН777». Добавленная запись в таблицу 5.2 добавится и в соответствующую таблицу БД.

Фактический результат: совпало с ожидаемым результатом.

**Тест 6.** Обновление записи в связанной таблице.

Суть теста: проверить, что данные в столбце таблицы с выпадающим списком обновятся, при обновлении записи в связанной таблице.

Шаги тестирования:

* Запустить приложение.
* Изменить в таблице 5.2 запись:

|  |  |  |
| --- | --- | --- |
| 2 | 2 | Б001ВК777 |

* Проверить, что в таблице 5.1 столбца «Гос. Номер» у выпадающего списка обновился список элементов.
* Проверить, что данные обновились и в БД.
* Ожидаемый результат: в таблице 5.1 у столбца «Гос. Номер» с выпадающим списком обновится список элементов. В частности, вариант выбора «Б526ВК777» изменится на «Б001ВК777». Если элемент «Б526ВК777» был выбран в качестве элемента по умолчанию в других записях, то он будет соответственно заменен на «Б001ВК777»

Фактический результат: совпало с ожидаемым результатом.