

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА



Институт радиоэлектроники и информационных технологий

ОТЧЕТ

по практической работе №6

«Работа с автономной частью клиента. Часть 2»

по дисциплине

Базы данных

РУКОВОДИТЕЛЬ:

(подпись)

профессор каф. ВСТ
Мисевич П. В.
(фамилия, и.,о.)

СТУДЕНТ:

(подпись)

Сухоруков В.А.
(фамилия, и.,о.)
19-В-2
(шифр группы)

Работа защищена «__» _____

С оценкой _____

Нижний Новгород 2021

Цель

Изучить способы работы с Объектами DataColumn и Constraint.

Ход выполнения

Свойства объекта DataColumn

1. Readonly - возвращает или задаёт значение, указывающее на допустимость изменения столбца после добавления строки в таблицу.

```
DataTable table = new DataTable();

DataColumn column = table.Columns.Add("ReadonlyColumn", typeof(string));
column.ReadOnly = true; // столбец таблицы с именем ReadonlyColumn доступен только для чтения

DataRow newRow = table.NewRow();

newRow[0] = "ReadonlyValue";

table.Rows.Add(newRow);

Console.WriteLine(table.Rows[0][0]);

table.Rows[0][0] = "NewValue"; // ОШИБКА времени выполнения
```

2. AllowDBNull - возвращает или задаёт значение, указывающее на допустимость нулевых значений в этом столбце для строк, принадлежащих таблице.

```
DataTable table = new DataTable();

DataColumn column = table.Columns.Add("AllowDBNullColumn", typeof(int));
column.AllowDBNull = false; // Запрет добавления нулевых значений |

DataRow newRow = table.NewRow();

newRow[0] = DBNull.Value;

table.Rows.Add(newRow); //ошибка времени выполнения

Console.WriteLine(table.Rows[0][0]);
```

3. `MaxLength` - возвращает или задаёт максимальную длину текстового столбца.

```
DataTable table = new DataTable();

DataColumn column = table.Columns.Add("MaxLengthConstraintColumn", typeof(string));
column.MaxLength = 5;

DataRow newRow = table.NewRow();

newRow[0] = "Some value";

table.Rows.Add(newRow); // ошибка времени выполнения

Console.WriteLine(table.Rows[0][0]);
```

4. `Unique` - возвращает или задаёт значение, показывающее должны ли значения в каждой строке быть уникальными.

```
DataTable table = new DataTable();

DataColumn column = table.Columns.Add("UniqueColumn", typeof(string));
column.Unique = true;

DataRow newRow = table.NewRow();
newRow[0] = "NonUniqueValue";
table.Rows.Add(newRow);

newRow = table.NewRow();
newRow[0] = "NonUniqueValue";
table.Rows.Add(newRow); // ошибка времени выполнения при нарушении ограничения Unique

Console.WriteLine(table.Rows[0][0]);
Console.WriteLine(table.Rows[1][0]);
```

Ограничения, добавляемые для построения таблиц. Метод Constraint.

1. Класс UniqueConstrain предоставляет ограничение на набор столбцов, в которых все значения должны быть уникальными.

```
// Класс UniqueConstraint предоставляет ограничение на набор столбцов, в которых все значения должны быть уникальными.
// Следует пользоваться этим ограничением в том случае, когда необходимо гарантировать уникальность
// комбинаций значений различных полей таблицы

namespace CBS.ADO_NET.TableConstraints
{
    static class TableExtentions
    {
        //Метод добавления строки с двумя столбцами
        public static void AddRow(this DataTable table, string column1Val, string column2Val)
        {
            var newRow = table.NewRow();

            newRow[0] = column1Val;
            newRow[1] = column2Val;

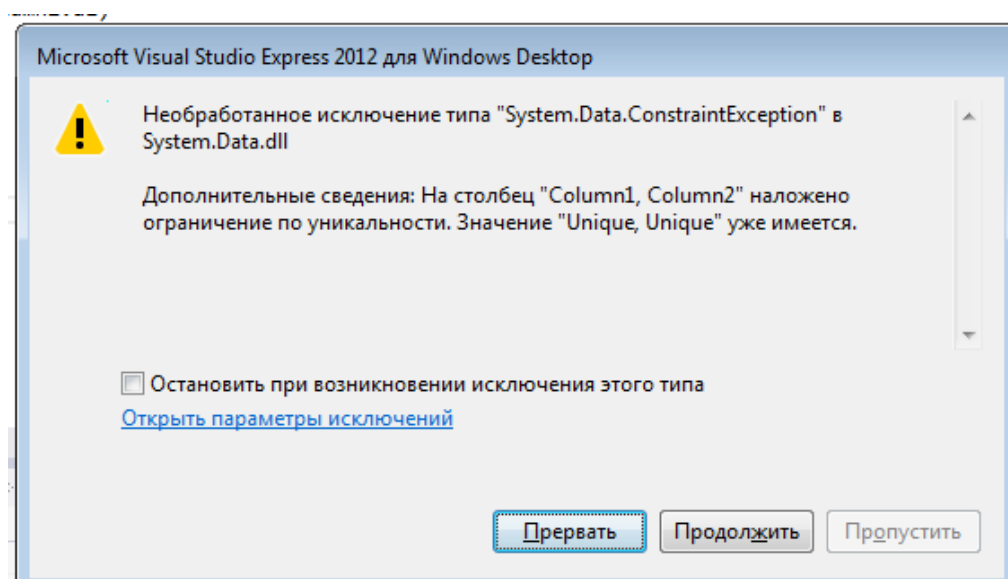
            table.Rows.Add(newRow);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            DataTable table = new DataTable();

            DataColumn column1 = table.Columns.Add("Column1", typeof(string));
            DataColumn column2 = table.Columns.Add("Column2", typeof(string));

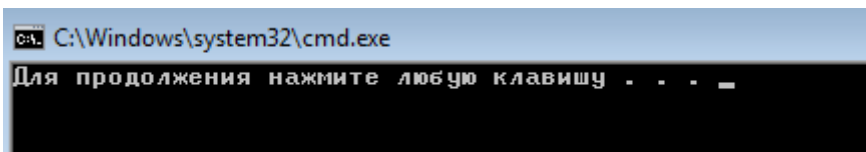
            //Добавление свойства UniqueConstrain
            table.Constraints.Add("tableUniqueConstraint", new[] { column1, column2 }, false);

            table.AddRow("Unique", "Unique");
            table.AddRow("Unique", "Unique");// ошибка времени выполнения строку со значениями Unique, Unique добавить нельзя
        }
    }
}
```



Если добавлять строки с разными значениями, то ошибки не возникает.

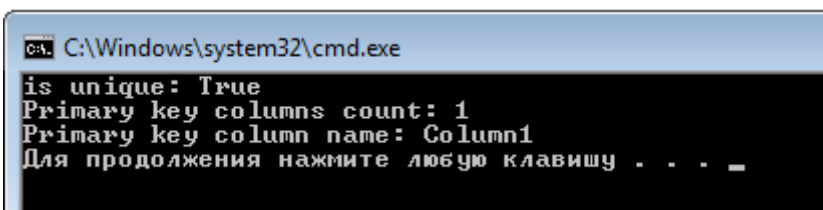
```
table.AddRow("Вася", "Петя");  
table.AddRow("Вася", "Вася");//Ошибка не возникает
```



```
C:\Windows\system32\cmd.exe  
Для продолжения нажмите любую клавишу . . . _
```

2. Используя свойство UniqueConstrain можно сделать столбец столбцом уникальных первичных ключей.

```
DataTable table = new DataTable();  
  
DataColumn column1 = table.Columns.Add("Column1", typeof(string));  
DataColumn column2 = table.Columns.Add("Column2", typeof(string));  
  
//Добавление свойства уникальности первому столбцу  
table.Constraints.Add(new UniqueConstraint(column1, true));  
  
Console.WriteLine("is unique: " + table.Columns[0].Unique);  
Console.WriteLine("Primary key columns count: " + table.PrimaryKey.Length);  
  
if (table.PrimaryKey.Length != 0)  
    Console.WriteLine("Primary key column name: " + table.PrimaryKey[0].ColumnName);  
else  
    Console.WriteLine("Primary key column name: No data");
```



```
C:\Windows\system32\cmd.exe  
is unique: True  
Primary key columns count: 1  
Primary key column name: Column1  
Для продолжения нажмите любую клавишу . . . _
```

3. `ForeignKeyConstrain` – ограничение, гарантирующее что нельзя создать строку в дочерней таблице, которая ссылается на несуществующую строку родительской таблицы.

```
DataSet ds = new DataSet(); // создание DataSet

DataTable parentTable = new DataTable(); // родительская таблица
DataTable childTable = new DataTable(); // дочерняя таблица

DataColumn childColumn = childTable.Columns.Add("ChildColumn", typeof(int));
DataColumn parentColumn = parentTable.Columns.Add("ParentColumn", typeof(int));

// ограничение ForeignKeyConstraint будет работать если родительская и дочерняя
//таблица находятся в одном объекте DataSet
ds.Tables.AddRange(new DataTable[] { childTable, parentTable });

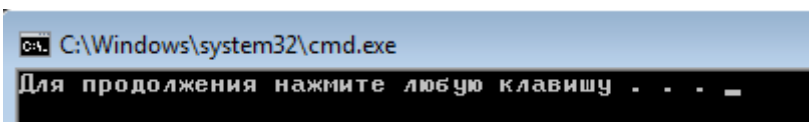
parentTable.Constraints.Add(new UniqueConstraint(parentColumn));
childTable.Constraints.Add(new ForeignKeyConstraint(parentColumn, childColumn));

DataRow parentRow = parentTable.NewRow();
parentRow[0] = 1;
parentTable.Rows.Add(parentRow);

DataRow childRow = childTable.NewRow();
childRow[0] = 1;

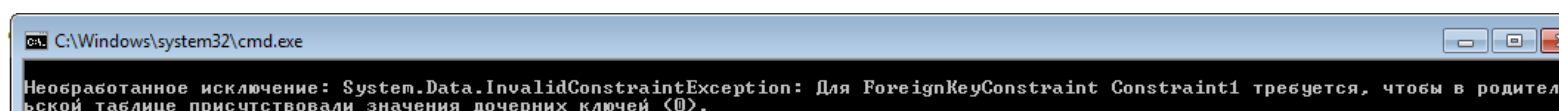
// после создания ограничения ForeignKeyConstraint нельзя добавлять в дочернюю таблицу строку
//, ссылающиеся на несуществующие строки из родительской таблицы
// childRow[0] = 0;
childTable.Rows.Add(childRow);
```

При совпадении ключей ошибок не возникает.



При изменении ключа ошибка возникает.

```
// после создания ограничения ForeignKeyConstraint нельзя добавлять в дочернюю таблицу строку
//, ссылающиеся на несуществующие строки из родительской таблицы
childRow[0] = 0;
childTable.Rows.Add(childRow);
```



4. Используя свойства метода Constraint можно создать ограничения на связь между таблицами. Рассмотрим, как это можно реализовать на примере базы данных ShopDB.

```
string connectionString = @"Data Source=.\SQLEXPRESS; Initial Catalog=ShopDB; Integrated Security=True;"; // создание строки подключения

DataSet ds = new DataSet();
DataTable customers = new DataTable("Customers");
DataTable orders = new DataTable("Orders");

using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();

    SqlCommand customersCmd = new SqlCommand("SELECT CustomerNo, LName, FName, Address1, Phone FROM Customers", connection);
    SqlCommand ordersCmd = new SqlCommand("SELECT OrderID, CustomerNo, OrderDate FROM Orders", connection);

    SqlDataReader ordersReader = ordersCmd.ExecuteReader(); // получение DataReader для таблицы OrderDetails

    // метод LoadWithSchema позволяет на основе объекта DataReader создать объект DataTable
    // с ограничениями для столбцов как в базе данных и заполнить эту таблицу данными
    orders.LoadWithSchema(ordersReader);
    ordersReader.Close();

    SqlDataReader customersReader = customersCmd.ExecuteReader();
    customers.LoadWithSchema(customersReader);
    customersReader.Close();
}

// объект DataReader не имеет информации об ограничениях объектов DataTable, таких как
// UniqueConstraint, ForeignKeyConstraint и PrimaryKey, поэтому придется их создать вручную
customers.PrimaryKey = new DataColumn[] { customers.Columns[0] };

ds.Tables.AddRange(new DataTable[] { customers, orders });

// создание ограничения ForeignKeyConstraint для таблицы OrderDetails
var FK_CustomersOrders = new ForeignKeyConstraint(customers.Columns["CustomerNo"], orders.Columns["CustomerNo"]);
orders.Constraints.Add(FK_CustomersOrders);

parentGridView.DataSource = customers; // связывание элемента управления parentGridView с таблицей Products
childDataGridView.DataSource = orders; // Связывание элемента управления childDataGridView с таблицей OrderDetails
```

	CustomerNo	LName	FName	Address1	Phone
▶	1	Круковский	Иван*+++++	Лужная 15	3222233322
	2	Дурнев	Виктор	Зелинская 10	(067)4242132
	3	Унакий	Зигмунд	Дихтяревская 5	(092)7612343
	4	Левченко	Виталий	Глушенка 5	(053)3456788
	5	Выжлецов	Олег	Киевская 3	(044)2134212
	20	Андреев	Николай	Лужная 9	(099)4578596

	OrderID	CustomerNo	OrderDate
▶	1	1	28.12.2009
	2	3	01.09.2010
	3	5	18.09.2010
	4	1	10.12.2010
*			

При изменении данных в форме в таблице Orders можно проверить работоспособность программы. Если номер клиента изменить на существующий, то программа продолжит работу, на несуществующий – выдаст ошибку.

	CustomerNo	LName	FName	Address1	Phone
▶	1	Круковский	Иван*+++++	Лужная 15	3222233322
	2	Дурнев	Виктор	Зелинская 10	(067)4242132
	3	Унакий	Зигмунд	Дихтяревская 5	(092)7612343
	4	Левченко	Виталий	Глушенка 5	(053)3456788
	5	Выжлецов	Олег	Киевская 3	(044)2134212
	20	Андреев	Николай	Лужная 9	(099)4578596

	OrderID	CustomerNo	OrderDate
	1	20	28.12.2009
▶	2	3	01.09.2010
	3	5	18.09.2010
	4	1	10.12.2010
*			

Окно сообщения об ошибке DataGridView по умолчанию

Исключение в DataGridView:

System.Data.InvalidConstraintException: Для ForeignKeyConstraint Constraint1 требуется, чтобы в родительской таблице присутствовали значения дочерних ключей (6).
в System.Data.ForeignKeyConstraint.CheckConstraint(DataRow childRow, DataRowAction action)
в System.Data.DataTable.RaiseRowChanging(DataRowChangeEventArgs args, DataRow eRow, DataRowAction eAction, Boolean fireEvent)
в System.Data.DataTable.SetNewRecordWorker(DataRow row, Int32 proposedRecord, DataRowAction action, Boolean isInMerge, Boolean suppressEnsurePropertyChanged, Int32 position, Boolean fireEvent, Exception& deferredException)
в System.Data.DataTable.SetNewRecord(DataRow row, Int32 proposedRecord, DataRowAction action, Boolean isInMerge, Boolean fireEvent, Boolean suppressEnsurePropertyChanged)
в System.Data.DataRow.EndEdit()
в System.Data.DataRowView.EndEdit()
в System.Windows.Forms.CurrencyManager.EndCurrentEdit()
в System.Windows.Forms.DataGridView.DataGridViewDataConnection.OnRowValidating(DataGridViewCellCancelEventArgs e)

Для замены этого окна по умолчанию обработайте событие DataError.

OK

Вывод

В ходе лабораторной работы были Изучены способы работы с объектами DataColumn и Constraint.