DOI: 10.18372/2410-7840.19.11440
УДК 004.658.2

МАСКИРОВАНИЕ ТАБЛИЦ БАЗЫ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ SQL CLR

Михаил Коломыцев, Светлана Носок, Анастасия Мазуренко

Статья посвящена актуальной проблеме защите информации в базах данных. Авторы рассматривают метод защиты данных путем маскирования. Суть маскирования данных состоит в необратимой замене в базе данных конфиденциальной информации (например, данных, идентифицирующих конкретных людей) несекретными данными для предотвращения доступа к ней неавторизованными пользователями. Как правило, конфиденциальные данные заменяются похожими на реальные значения, чтобы их можно было использовать в тестовых системах с гарантией, что первоначальные данные не могут быть получены, извлечены или восстановлены. Маскирование позволяет владельцам базы данных самим определять, сколько конфиденциальных данных следует отображать, при минимальном влиянии на работу приложений – данные должны оставаться функционально пригодными для прикладной обработки (в основном, задач тестирования, обучения и т.п.). В данной статье авторы предлагают методику маскирования персональных данных в базе данных (БД). Данная методика реализована в виде CLR–сборки для СУБД MS SQL Server.

Ключевые слова: *база данных, защита персональных данных, маскирование данных, конфиденциальные данные, информационная система.*

АКТУАЛЬНОСТЬ И ПОСТАНОВКА ЗАДАЧИ МАСКИРОВАНИЯ ДАННЫХ

Практика разработки информационных систем показывает, что кроме производственной (основной) базы данных возникает задача создания ее копий, непроизводственных (тестовых) баз данных. Связано это с рядом причин:

– необходимостью проведения работ по развитию информационной системы, ее тестированию, обучению пользователей;

– копии БД могут использоваться для аналитической обработки (data mining);

– предоставление данных партнерам в рамках производственной или торговой кооперации и других целей.

По некоторым оценкам, число непроизводственных баз данных может достигать 6-8 экземпляров. И, если в производственной БД поддерживается адекватная политика безопасности, то в тестовых БД вопросам безопасности внимания уделяется меньше, что повышает риски утечки данных. Типичными случаями является бесконтрольное создание новых учетных записей в тестовой БД и извлечение из нее данных для других приложений.

Согласно данных Oracle [1], почти половина компаний, принимавших участие в проведенном обследованиях, заявляли, что не контролируют возможность потери данных из тестовых БД.

Особую актуальность в Украине приобрел вопрос защиты персональных данных, после принятия соответствующего закона [2]. Согласно закону, владельцы баз персональных данных обязаны обеспечить их защиту.

Для защиты данных в такой ситуации можно использовать подход, который называется маскирование данных [3]. Маскирование данных представляет собой процесс обезличивания или сокрытия определенных данных в таблицах базы. Маскирование данных по своей сути защищает конфиденциальные данные от неавторизованного доступа, изменяя их значения, но сохраняя при этом первоначальные ограничения на значения данных.

Важной особенностью непроизводственных БД является то, что они должны быть функционально эквивалентными производственной БД. Это означает, что для обработки информации в тестовой БД можно использовать те же приложения, что и для основной БД.

Постановка задачи. Рассмотрим ситуацию, когда существует таблица БД, с колонками, типы данных которых, относятся к базовым типам – строка, число, дата. Необходимо построить высокопроизводительную процедуру маскирования информации в таблице. Маскированные данные должны отвечать требованию сохранения формата данных.

В данной статье авторы предлагают методику статического маскирования данных с использованием технологии SQL CLR. Данная методика реализована в виде сборки для СУБД MS SQL Server.

Целью работы является разработка методики статического маскирования данных с помощью высокопроизводительных функций на языке C#.

ТРЕБОВАНИЯ К МЕТОДАМ МАСКИРОВАНИЯ

К типичным ограничениям процесса маскирования данных можно отнести [3, 5]:

Невозможность восстановления исходных данных. Это очевидное условие, без которого маскирование теряет смысл. При реализации процесса маскирования необходимо учитывать, что результат преобразования не должен быть обратимым в том смысле, что у лиц, не имеющих доступа к ключевой информации, не должно быть возможности восстановить оригинальный текст, используя маскированный.

Маскированные данные должны сохранять принадлежности к определенному домену. Данное ограничение можно представить в виде набора частных требований, например:

– **Сохранение типа данных.** Результаты маскирования должны относиться к тем же базовым типам данных, что и исходные данные.

– **Сохранение формата.** Маскированные данные должны иметь такую же структуру, что и исходные данные. Это означает, что, если исходные данные имеют, например, размер от 2 до 30 символов, маскированные данные должны также отвечать этому условию. В предлагаемой методике требование подразумевает так же:

– сохранение регистра символов,

– сохранение длины строковых переменных,

– нахождение гласных и согласных букв на тех же позициях, что и в исходной строке,

– возможность оставлять без изменений некоторые символы или подстроки исходной строки.

Важными являются так же свойства:

– Применимость алгоритма маскирования ко всему множеству данных домена.

– Маскирование должно быть автоматизированным, легко повторяемым процессом.

ВЫБОР МЕТОДА МАСКИРОВАНИЯ

Анализ существующих методов маскирования [3, 4, 6] позволяет сделать вывод о их достоинствах и недостатках. Широко используемые методы, такие как: замена на символ-константу, подстановка на основе таблицы подстановок, перестановка не удовлетворяют в силу их ненадежности. Шифрование с сохранение формата является ресурсоемкой операцией.

По способу организации процесса маскирования различают статическое и динамическое маскирование. При статическом маскировании создается копия производственной БД с заменой защищаемых данных на маскированные значения. Статическое маскирование позволяет создать реалистичные тестовые БД и снизить риск раскрытия информации в непроизводственной среде, поскольку тестовая БД не содержит немаскированных данных.

Авторы предлагают методику статического маскирования путем посимвольной замены исходных данных на случайные значения, совпадающий с исходными значениями по определенным признакам. Сохранность признаков позволяет говорить о том, что маскированные данные формально принадлежат к тому же домену, что и исходные данные. Это следующие признаки:

– замена алфавитного символа на алфавитный, принадлежащий к той же кодовой странице;

– сохранение регистра алфавитных символов;

– замена цифр цифрами;

– сохранение при необходимости некоторых символов или диапазона символов неизмененными.

Последнее требование важно для обеспечения сохранности домена для таких данных, как почтовые адреса, знаки денежных единиц, знаки разделителей в числах, сохранение первых трех цифр телефонного номера и др.

РАЗРАБОТКА ПРОЦЕДУРЫ МАСКИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ SQL CLR

Предлагаемая авторами методика предназначена для реализации в MS SQL Server. Функционал T-SQL очень широк и для большинства операций с данными хватает его возможностей. Но иногда необходимо выполнить над данными в базе данных такие операции, для которых не хватает функционала стандартных средств SQL Server. Одним из вариантов их решения является использование технологии SQL CLR – возможность создавать CLR функции/процедуры, путем подключения самостоятельно написанных библиотек, с реализованным функционалом, например, с помощью Visual Studio на C#. Классический пример такой задачи - регулярные выражения. Кроме того, ряд задач SQL CLR может выполнить быстрее чем T-SQL, например, сложные вычисления [7].

Методика маскирования состоит из следующих этапов:

a. В среде Visual Studio создается библиотека, реализующая функционал маскирования.

b. В MS SQL Server создается сборка для подключения библиотеки и пользовательские функции, вызывающие функции маскирования из библиотеки.

с. Выполняется маскирование исходной таблицы.

Ниже приводится текст процедуры маскирование на языке C#, реализующей данную методику. Основной является функция MaskString, выполняющей маскирование строки. Параметрами функции являются: исходная строка, список немаскируемых символов, начальная и конечная позиции в исходной строке, в пределах которых строка не маскируется. Функция возвращает маскированную строку.

Функция MaskMoney демонстрирует маскирование числовых данных, а функция MaskDate – маскирование даты.

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.Linq;
using System.Globalization;

public partial class UserDefinedFunctions
{
  [Microsoft.SqlServer.Server.SqlFunction]
  public static String Substitution_Pattern(Char[] Pat, char c, Int32 Nmax, Int32 Nbase, Int32 n)
    {
        string rr = string.Empty;
        int dlt = 0;
         if (Pat.Contains(c))
        { dlt = c - System.Char.ToUpper(c);
          if ((Nmax>0) && (c - System.Char.ToUpper(c)) == Nmax)
            { n += Nbase; }
          try
             { rr += Pat[n];}
          catch (Exception ex)
             {throw new Exception("\n n=" + n + " dlt=" + dlt + " c=" + c + "\n" + ex.Message);}
         }
        return rr;
     }

    public static SqlString MaskString(SqlString SourceString, SqlString NoChangeString, Int32
StartNoChange, Int32 EndNoChange)
    {
        char[] vowelsEn = "AEIOUaeiou".ToCharArray();
        char[] consonantsEn = "BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz".ToCharArray();
        char[] consonantsCr = "БВГҐДЖЗКЛМНПРСТФХЦЧШЩЬьбвгґджзклмнпрстфхцчшщьь".ToCharArray();
        char[] vowelsCr = "АЕЄЁИІЙЇОУЫЭЮЯаеєёиіїйоуыэюя".ToCharArray();
        char[] digits = "0123456789".ToCharArray();
        char[] src_array;
        char[] nochange_array;
        int seed = (int)DateTime.Now.Ticks % 10000000;
        int i = 0, len = 0, n=0;
        string rtn = string.Empty;
        try
        {
            if (SourceString.IsNull) {return new SqlString();}
            src_array = SourceString.Value.ToCharArray();
            nochange_array = NoChangeString.Value.ToCharArray();
            len = src_array.Length;
             for (i = 0; i <= len - 1; i++)
            {
                char c = src_array[i];
                if ((i >= StartNoChange && i <= EndNoChange) || (nochange_array.Contains(c)))
                {
                    rtn += c;
                    continue;
                }
                else
                {
```

```
                seed = (int)(DateTime.UtcNow.Ticks % 1000000000) + i;
                Random rdm = new Random(Guid.NewGuid().GetHashCode());
                n = rdm.Next(seed) % 5;
                rtn += Substitution_Pattern(vowelsEn, c, 32, 5, n);
                n = rdm.Next(seed) % 21;
                rtn += Substitution_Pattern(consonantsEn, c, 32, 21, n);
                n = rdm.Next(seed) % 10;
                rtn += Substitution_Pattern(vowelsCr, c, 32, 14, n);
                n = rdm.Next(seed) % 22;
                rtn += Substitution_Pattern(consonantsCr, c, 32, 23, n);
                n = rdm.Next(seed) % 10;
                rtn += Substitution_Pattern(digits, c, 0, 10, n);
            }
        }
    }
    catch (Exception ex)
    {
        throw new Exception("\n\n" + ex.Message);
    }

    return new SqlString(rtn);
}

public static SqlMoney MaskMoney(SqlMoney SourceVal)
{
    string mon_array, monval;
    SqlString rtm = new SqlString();
    decimal drtn = 0;
    if (SourceVal.IsNull) { return new SqlMoney(); }
    mon_array = SourceVal.ToString();
    rtm = MaskString(new SqlString(mon_array), "., ", 0, -1);
    monval = rtm.ToString();
    try
    {
        drtn = Convert.ToDecimal(monval);
    }
    catch (System.OverflowException)
    {
        System.Console.WriteLine("The conversion from string to decimal overflowed.");
    }
    catch (System.FormatException)
    {
        System.Console.WriteLine("The string is not formatted as a decimal.");
    }
    return new SqlMoney(drtn);
}

public static SqlDateTime MaskDate(SqlDateTime SourceVal)
{
    string dateval, ret_dt;
    SqlString sqlYear = new SqlString();
    SqlString sqlMonth = new SqlString();
    SqlString sqlDay = new SqlString();
    string sYear, sMonth, sDay;
    int iYear = 0, iMonth=0, iDay=0;

    DateTime myDate = new DateTime(2015, 1, 1, 1, 1, 1);

    if (SourceVal.IsNull) { return new SqlDateTime(); }

    var culture = new CultureInfo("en-US");
    DateTime date =  (DateTime) SourceVal;
    dateval = date.ToString("s");

    sqlYear = MaskString(new SqlString(dateval.Substring(0, 4)), "., ", 0, -1);
```

```
        sqlMonth = MaskString(new SqlString(dateval.Substring(5, 2)), ".", 0, -1);
        sqlDay = MaskString(new SqlString(dateval.Substring(8, 2)), ".", 0, -1);
        sYear = sqlYear.ToString();
        sMonth = sqlMonth.ToString();
        sDay = sqlDay.ToString();

        iYear = Convert.ToInt32(sYear) % 9999;
        if (iYear < 1753) { iYear = 1753; }
        sYear = iYear.ToString();

        iMonth = Convert.ToInt32(sMonth) % 12;
        if (iMonth == 0) { sMonth = "01"; }
        else if (iMonth < 10) {sMonth = "0"+iMonth.ToString();}
        else       {sMonth = iMonth.ToString(); }

        iDay = Convert.ToInt32(sDay) % 28;
        if (iDay == 0) { sDay = "01"; }
        else if (iDay < 10) {sDay = "0"+iDay.ToString();}
        else       {sDay = iDay.ToString(); }

        ret_dt = sYear + "-" + sMonth + "-" + sDay ;
        myDate = DateTime.ParseExact(ret_dt, "yyyy-MM-dd", CultureInfo.InvariantCulture);
        return  (SqlDateTime) (myDate);
    }

}
```

Создание сборки и пользовательских функций в MS SQL Server выполняется такими командами:

```
use Test; -- наша БД
alter database Test set trustworthy on;
go
exec sp_configure 'clr enabled', 1;
reconfigure;
go
CREATE ASSEMBLY [clrFunction]  AUTHORIZATION [dbo]
from 'C:\Users\Kol\Documents\Visual Studio 2010\Projects\DataMasking\DataMasking\bin\Debug\DataMasking.dll' --путь к dll
--WITH PERMISSION_SET = UNSAFE ;
go
CREATE FUNCTION [dbo].[fn_MaskString] (@SourceString nvarchar(max), @NoChangeString nvarchar(max), @StartNoChange int =0, @EndNoChange int=-1)
RETURNS nvarchar(max)
AS EXTERNAL NAME [clrFunction].[UserDefinedFunctions].[MaskString];
go
CREATE FUNCTION [dbo].[fn_MaskMoney] (@SourceVal money)
RETURNS money
AS EXTERNAL NAME [clrFunction].[UserDefinedFunctions].[MaskMoney];
go
CREATE FUNCTION [dbo].[fn_MaskDate] (@SourceVal DateTime)
RETURNS DateTime
AS EXTERNAL NAME [clrFunction].[UserDefinedFunctions].[MaskDate];
go
```

Рассмотрим пример. На рисунке 1 приведена исходная таблица. Первые четыре столбца имеют строковый тип (varchar), пятый – числовой (money), шестой – дата (datetime).

	ФИО	ИНН	Почта	Адрес	Кредит карта	Баланс	Срок действия
1	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.547
2	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.547
3	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.547
4	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.547
5	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.547
6	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.547
7	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.547
8	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.547
9	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.550
10	Петренко П.І.	0123456789	petrenko@gmail.com	Одеса, вул. Буніна, 99	01234 1234 2345 3456	1000,00	2017-01-08 11:58:10.550

Рис. 1. Исходная таблица с немаскированными данными

Все строки таблицы одинаковые, для демонстрации вариативности процесса маскирования.

Маскирование выполним с помощью T-SQL:

```
use Test
update dbo.EmpData
set [Name]=dbo.fn_MaskString(Name,'!:,. ', default, default)
, [INN]= dbo.fn_MaskString(INN,'!:,. ', default, default)
, [Email] = dbo.fn_MaskString(Email, '_@., ', default, default)
, [Adress] = dbo.fn_MaskString(Adress, '_#№., ', default, default)
, [CreditCard] = dbo.fn_MaskString(CreditCard,'.:, ', default, default)
, [Balance] = dbo.fn_MaskMoney(Balance)
, [ExpirDate] = dbo.fn_MaskDate(ExpirDate); .
```

Маскированная таблица изображена на рис. 2.

	ФИО	ИНН	Почта	Адрес	Кредит карта	Баланс	Срок действия
1	Дуккефгу П.О.	3127217273	zustubyi@rjeuw.xiw	Єніъї, щїр. ТирЇлё, 88	10964 4406 6696 9180	544,57	8971-10-06 00:00:
2	Вйбчендй Ш.И.	4197952415	qabwutni@lwiuq.den	Єнело, вїф. Цjзєли, 93	13476 0501 3515 4461	2688,61	1753-11-20 00:00:
3	Нёщжікди Т.У.	2649869350	quprimwo@mvioc.ror	Йшіфе, кем. МанУза, 19	99029 5717 6114 9508	2807,64	2179-11-15 00:00:
4	Шіжвибрї Ж.Ї.	0035976257	wulqehla@jraof.qit	Ёжупи, фєц. ЛёпЄси, 42	48470 0471 9788 9679	1501,81	4410-05-16 00:00:
5	Віхпічбї Ф.Ї.	0960690375	poyyupgo@pxuaw.nuw	Єдозё, шєщ. ЪогІшо, 78	22700 7771 0021 3805	8028,37	3951-02-25 00:00:
6	Гётъётчё М.І.	3547242765	felhacgo@zsaeg.nuk	Урежи, гиш. ЪипЙто, 14	63856 3457 1419 2159	9758,71	2299-02-14 00:00:
7	Феврецсї Ш.И.	5124997791	hanvahpo@qdoin.lom	Ётйзй, шуф. БёшЕно, 71	51712 5674 9906 7733	9618,15	5306-10-27 00:00:
8	Роцрёпцє Ш.І.	4531608181	qeckabja@dpeom.pis	Їлижи, ъиъ. БущИші, 97	97114 8204 5043 3562	6239,65	8088-08-23 00:00:
9	Хидгїґті Ъ.Й.	6328386618	qotnuwyi@lbuoy.xem	Одёфа, рїф. РапЄщї, 01	52252 8378 4548 3009	212,10	3535-10-09 00:00:
10	Бенцоччй Г.И.	1535915908	gicmapfu@zpeul.kap	Ичепо, мёт. НёгЕва, 57	88936 5210 9604 1828	654,39	8826-01-01 00:00:

Рис. 2. Таблица с маскированными данными

ЗАКЛЮЧЕНИЕ

Предложенная авторами методика позволяет получить маскированные данные, обработка которых может выполняться теми же приложениями, что и используемые в производственной базе данных. Использование технологии SQL CLR позволяет сделать процедуру маскирования высокоскоростной. Необратимая замена данных в процессе маскирования гарантирует конфиденциальность данных. Преобразованные по изложенной методике данные могут использоваться в непроизводственных БД для таких целей, как тестирования приложений и обучения персонала.

ЛИТЕРАТУРА

[1]. Ahmed W. Data Masking Best Practice [Электронный ресурс] / W. Ahmed, J. Athreya. – 2013. – Режим доступа: http://www.oracle.com/ us/products/database/data-masking-best-practices-161213.pdf.

[2]. Закон України «Про захист персональних даних» від 20.12.2012 №2297- VI.

[3]. Коломыцев М.В., Южаков А.М. Защита персональных данных методом маскирования / М. В. Коломыцев, А. М. Южаков // Захист інформації. - 2013. - Т. 15, № 4. - С. 382-387. - Режим доступа: http://nbuv.gov.ua/j-pdf/Zi_2013_15_4_17.pdf.

[4]. Understanding and Selecting Data Masking Solutions: Creating Secure and Useful Data [Электронный ресурс]. – 2012. – Режим доступа: https://securosis.com/assets/library/reports/UnderstandingMasking_FinalMaster_V3.pdf.

[5]. The Five Laws Of Data Masking [Электронный ресурс]. – 2008. – Режим доступа: https://securosis.com/blog/the-five-laws-of-data-masking.

[6]. Коломыцев М.В., Носок С.А., Мазуренко А.Е. Обеспечение целостности внешних ключей маскированной базы данных // Захист інформації – 2015. – Т.17, № 4. - С.306-311.
[7]. S. Rutzky CLR Performance Testing [Электронный ресурс] https://www.simple-talk.com/sql/t-sql-programming/clr-performance-testing/.

REFERENCES

[1]. Ahmed W. Data Masking Best Practice [Electronic source] / W. Ahmed, J. Athreya. – 2013. – Access mode: http://www.oracle.com/us/products/database/data-masking-best-practices-161213.pdf.
[2]. The Law of Ukraine "About personal data protection" from 20.12.2012 №2297- VI.
[3]. M. Kolomytsev, A. YuzhakovPersonal data protection using masking technique / M. Kolomytsev, A. Yuzhakov // Information protection. - 2013. - T. 15, № 4. - P. 382-387. – Access mode: http: //nbuv.gov.ua/j-pdf/Zi_2013_15_4_17.pdf.
[4]. Understanding and Selecting Data Masking Solutions: Creating Secure and Useful Data [Electronic source]. – 2012. – Access mode: https://securosis.com/assets/library/reports/UnderstandingMasking_Final-Master_V3.pdf.
[5]. The Five Laws Of Data Masking [Electronic source]. – 2008. – Access mode: https://securosis.com/blog/the-five-laws-of-data-masking.
[6]. M. Kolomytsev, S. Nosok, A. Mazurenko Ensuring the foreign key integrity of the masked database // Information protection – 2015. – T.17, № 4. - P.306-311.
[7]. S. Rutzky CLR Performance Testing [Electronic source] https://www.simple-talk.com/sql/t-sql-programming/clr-performance-testing/.

МАСКУВАННЯ ТАБЛИЦЬ БАЗИ ДАНИХ ІЗ ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ SQL CLR

Стаття присвячена актуальній проблемі захисту інформації в базах даних. Автори розглядають метод захисту даних шляхом маскування. Суть маскування даних полягає у незворотній заміні в базі даних конфіденційної інформації (наприклад, даних, що ідентифікують конкретних людей) несекретними даними для запобігання доступу до неї неавторизованими користувачами. Як правило, конфіденційні дані замінюються схожими на реальні значення, щоб їх можна було використовувати в тестових системах з гарантією, що початкові дані не можуть бути отримані, витягнуті або відновлені. Маскування дозволяє власникам бази даних самим визначати, скільки конфіденційних даних слід відображати, при мінімальному впливі на роботу додатків – дані повинні залишатися функціонально придатними для прикладної обробки (в основному, завдань тестування, навчання тощо). В даній статті автори пропонують методику маскування персональних даних в базі даних (БД). Дана методика реалізована у вигляді CLR–складання для СУБД MS SQL Server.

Ключові слова: база даних, захист персональних даних, маскування даних, конфіденційні дані, інформаційна система.

DATABASE TABLES MASKING USING THE SQL CLR TECHNOLOGY

The article is devoted to the trending problem of the information protection in databases. The authors considering the data protection using masking method. The point of the data masking is in irreversible replacing confidential information in the database (e.g., data that identifying specific individuals) with unclassified data to prevent access by unauthorized users. As usual, the confidential data being replaced with values that seems like real, so they can be used in test systems with guarantee that the original data can not be retrieved, recovered or restored. Masking enables database owners to determine how much sensitive data to be displayed, with minimal impact on an application workflow - data must remain functionally suitable for application processing (mainly in testing and learning tasks, etc.). In this article, the authors propose a method of masking the personal data in the database (DB). This method is implemented as a CLR-build for DBMS MS SQL Server.

Keywords: database, personal data protection, data masking, confidential data, information system.

Коломиец Михайло Володимирович, кандидат технічних наук, доцент Фізико-технічного інституту НТУУ «КПІ».
E-mail: box144a@ukr.net
Коломыцев Михаил Владимирович, кандидат технических наук, доцент Физико-технического института НТУУ «КПИ».
Kolomytsev Myhailo, candidate of technical sciences, associate professor of Institute of Physics and Technologies of the NTUU "KPI".

Носок Світлана Олександрівна, кандидат технічних наук, доцент Фізико-технічного інституту НТУУ «КПІ».
E-mail: svetlana@pti.kpi.net
Носок Светлана Александровна, кандидат технических наук, доцент Физико-технического института НТУУ «КПИ».
Nosok Svitlana, candidate of technical sciences, associate professor of Institute of Physics and Technologies of the NTUU "KPI".

Мазуренко Анастасія Євгенівна, студентка Фізико-технічного інституту НТУУ «КПІ».
E-mail: ks0610@mail.ru
Мазуренко Анастасия Евгениевна, студентка Физико-технического института НТУУ «КПИ».
Mazurenko Anastasiia, student of the Institute of Physics and Technologies of the NTUU "KPI".