**Лабораторная № 2**

**«Шифрование в SQL Server»**

**Цель работы:**

Ознакомиться с шифрованием в SQL Server.

**Теоретическая информация:**

**Шифрование столбца данных с помощью простого симметричного шифрования**

# Основные понятия

**1. Главный ключ базы данных**

Главный ключ базы данных — это симметричный ключ, который применяется для защиты закрытых ключей сертификатов и асимметричный ключей, которые есть в базе данных. При создании этот главный ключ зашифровывается с помощью алгоритма AES\_256 и предоставленного пользователем пароля.

## [***Синтаксис***](javascript:void(0))

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'password'

## [***Аргументы***](javascript:void(0))

ПАРОЛЬ ='*password*'

Пароль, который использовался при шифровке главного ключа базы данных. *password* должен соответствовать требованиям политики паролей Windows применительно к компьютеру, на котором запущен экземпляр SQL Server.

**2. Сертификат**

Сертификат — это защищаемый объект уровня базы данных, соответствующий стандарту X.509 и поддерживающий поля X.509 V1.Инструкция CREATE CERTIFICATE может загрузить сертификат из файла или сборки. Она также может создать пару ключей и самостоятельно подписанный сертификат.

## [***Синтаксис***](javascript:void(0))

CREATE CERTIFICATE certificate\_name

WITH SUBJECT = 'certificate\_subject\_name';

## [***Аргументы***](javascript:void(0))

*certificate\_name-* Имя, под которым сертификат будет доступен в базе данных.

SUBJECT ='*certificate\_subject\_name*'

Термин *subject* (субъект) относится к полю в метаданных сертификата, определяемому стандартом X.509. Он может иметь длину до 128 символов. Субъекты, имеющие длину больше 128 символов, будут усечены, если они сохраняются в каталоге, однако в большом двоичном объекте (BLOB), содержащем сертификат, сохранится полное имя субъекта.

**3. Cимметричный ключ**

***Синтаксис***

CREATE SYMMETRIC KEY ‘key name’

WITH ALGORITHM = AES\_256

ENCRYPTION BY CERTIFICATE ‘sertificate name’;

[***Аргументы***](javascript:void(0))

*Key\_name*

Уникальное имя, под которым симметричный ключ известен в базе данных. Имена временных ключей должны начинаться с одиночного символа номера (#). Например, **#temporaryKey900007**. Нельзя создать симметричный ключ, в начале имени которого указано более одного символа #. Временный симметричный ключ невозможно создать с помощью поставщика расширенного управления ключами.

*certificate\_name*

Указывает имя сертификата, который будет использоваться для шифрования симметричного ключа. Сертификат уже должен существовать в базе данных.

**4. Зашифрование с помощью симметричного ключа**

EncryptByKey ( key\_GUID , { 'cleartext' | @cleartext }

    [, { add\_authenticator | @add\_authenticator }

     , { authenticator | @authenticator } ] )

## [***Аргументы***](javascript:void(0))

*key\_GUID*

Идентификатор GUID ключа, используемого для шифрования *cleartext*. **uniqueidentifier** .

'*cleartext*'

Данные, которые шифруются ключом.

@cleartext

Переменная типа **nvarchar**, **char**, **varchar**, **binary**, **varbinary** или **nchar**, содержащая данные, которые будут зашифрованы ключом.

*add\_authenticator*

Указывает, будут ли вместе с аргументом *cleartext* зашифрована структура проверки подлинности. При использовании структуры проверки подлинности аргумент должен иметь значение 1. **int** .

@add\_authenticator

Указывает, будут ли вместе с аргументом *cleartext* зашифрована структура проверки подлинности. При использовании структуры проверки подлинности аргумент должен иметь значение 1. **int** .

*authenticator*

Данные, на основе которых формируется структура проверки подлинности. **sysname** .

@authenticator

Переменная, содержащая данные, на основе которых формируется структура проверки подлинности.

## [***Типы возвращаемых данных***](javascript:void(0))

**varbinary** с максимальным размером 8000 байт.

## [***Замечания***](javascript:void(0))

Функция EncryptByKey использует симметричный ключ. Этот ключ должен быть открыт. Если симметричный ключ уже открыт в текущем сеансе, не нужно открывать его снова в контексте запроса.

## [***Пример***](javascript:void(0))

### Шифрование строки симметричным ключом

Следующий пример показывает, как добавить столбец к таблице Employee, а затем зашифровать значение номера социального страхования, который хранится в столбце NationalIDNumber.

ALTER TABLE HumanResources.Employee

ADD EncryptedNationalIDNumber varbinary(128);

GO

OPEN SYMMETRIC KEY SSN\_Key\_01

DECRYPTION BY CERTIFICATE HumanResources037;

UPDATE HumanResources.Employee

SET EncryptedNationalIDNumber

= EncryptByKey(Key\_GUID('SSN\_Key\_01'), NationalIDNumber);

GO

### 5. Расшифрование

## [***Синтаксис***](javascript:void(0))

DecryptByKey ( { 'ciphertext' | @ciphertext }

[ , add\_authenticator, { authenticator | @authenticator } ] )

## [***Аргументы***](javascript:void(0))

*ciphertext*

Данные, зашифрованные с помощью ключа. Аргумент *ciphertext* имеет тип varbinary.

**@ciphertext**

Переменная типа varbinary. Содержит данные, которые были зашифрованы с помощью ключа.

*add\_authenticator*

Указывает, было ли средство проверки подлинности зашифровано вместе с неформатированным текстом. Должно быть равно значению, переданному функции EncryptByKey при шифровании данных. Аргумент *add\_authenticator* имеет тип **int**

*authenticator*

Данные, для которых формируется средство проверки подлинности. Значение аргумента должно совпадать со значением, заданным функции EncryptByKey. Аргумент *authenticator* имеет тип **sysname**.

**@authenticator**

Переменная, содержащая сведения, из которых формируются данные для проверки подлинности. Значение аргумента должно совпадать со значением, заданным функции EncryptByKey.

## [***Типы возвращаемых данных***](javascript:void(0))

**varbinary** с максимальным размером 8 000 байт.

## [***Замечания***](javascript:void(0))

В функции DecryptByKey используется симметричный ключ. Этот симметричный ключ должен быть открыт уже в базе данных.Одновременно могут быть открыты несколько ключей. Открывать ключ непосредственно перед раскодированием необязательно.

Симметричное кодирование и декодирование осуществляется относительно быстро и подходит для работы с большими объемами данных.

## [***Пример***](javascript:void(0))

### Декодирование с помощью симметричного ключа

Следующий пример иллюстрирует декодирование зашифрованного текста с помощью симметричного ключа.

OPEN SYMMETRIC KEY SSN\_Key\_01

DECRYPTION BY CERTIFICATE HumanResources037;

GO

SELECT NationalIDNumber, EncryptedNationalID

AS 'Encrypted ID Number',

CONVERT(nvarchar, DecryptByKey(EncryptedNationalID))

AS 'Decrypted ID Number'

FROM HumanResources.Employee;

GO

**Задание:**

1. Установить SQL Server Express и SQL Management Studio

2. Создать базу данных UsersCreditCard

3. Создать таблицу CreditCard с двумя столбцами: CardID и CardNumber и заполнить ее произвольными значениями

4. Создать главный ключ базы

5. Создать сертификат

6. Создать симметричный ключ, алгоритм AES-256, зашифрованный с помощью созданного ранее сертификата

7. Создать новый столбец в таблице CreditCard, содержащий зашифрованные записи из столбца CardNumber.

8. Вывести запросом таблицу CreditCard и дополнительный столбец, расшифровывающий значение зашифрованного столбца.

9. Сравнить значения оригинальных значений столбца CardNumber и полученных в результате расшифровки.

10. По результатам работы предоставить отчет, содержащий ход выполнения работы, код запросов и скриншоты.

**Отчет необходимо оформить по шаблону с сайта «ЛЭТИ» (титульник, цель работы, ход выполнения работы, вывод).**

**Отчеты присылать на почту** [ZOC.leti@yandex.ru](https://vk.com/write?email=ZOC.leti@yandex.ru) **. В теме письма должно содержаться имя, фамилия, группа, номер лабораторной работы и слово «SQL Server».**