SQL Server decrypt stored procedures

```
/*==============================================================================

NAME:          Decrypt SQL 2005 stored procedures, functions, views, and triggers

DESCRIPTION:            HEADS UP: In order to run this script you must log in
                        to the server in DAC mode: To do so, type
                        ADMIN:<SQLInstanceName> as your server name and use the "sa"
                        or any other server admin user with the appropriate password.

                        CAUTION! DAC (dedicated admin access) will kick out all other
                        server users.

                        The script below accepts an object (schema name + object name)
                        that were created using the WITH ENCRYPTION option and returns
                        the decrypted script that creates the object. This script
                        is useful to decrypt stored procedures, views, functions,
                        and triggers that were created WITH ENCRYPTION.

                        The algorithm used below is the following:
                        1. Check that the object exists and that it is encrypted.
                        2. In order to decrypt the object, the script ALTER (!!!) it
                        and later restores the object to its original one. This is
                        required as part of the decryption process: The object
                        is altered to contain dummy text (the ALTER uses WITH ENCRYPTION)
                        and then compared to the CREATE statement of the same dummy
                        content.

                        Note: The object is altered in a transaction, which is rolled
                        back immediately after the object is changed to restore
                        all previous settings.

                        3. A XOR operation between the original binary stream of the
                        enrypted object with the binary representation of the dummy
                        object and the binary version of the object in clear-text
                        is used to decrypt the original object.


USER PARAMETERS:        @ObjectOwnerOrSchema
                        @ObjectName

RESULTSET:              NA

RESULTSET SORT:         NA

USING TABLES/VIEWS:     sys.sysobjvalues
                        syscomments

REVISIONS

DATE         DEVELOPER          DESCRIPTION OF REVISION             VERSION
=========    ===============    ================================    ===========
01/01/2007   Omri Bahat         Initial release                     1.00
==============================================================================*/

DECLARE @ObjectOwnerOrSchema NVARCHAR(128)
DECLARE @ObjectName NVARCHAR(128)
DECLARE @i INT
DECLARE @ObjectDataLength INT
DECLARE @ContentOfEncryptedObject NVARCHAR(MAX)
DECLARE @ContentOfDecryptedObject NVARCHAR(MAX)
DECLARE @ContentOfFakeObject NVARCHAR(MAX)
DECLARE @ContentOfFakeEncryptedObject NVARCHAR(MAX)
DECLARE @ObjectType NVARCHAR(128)
DECLARE @ObjectID INT

SET NOCOUNT ON
```

```
DECLARE ProcedureCursor CURSOR FOR
SELECT DISTINCT P.ROUTINE_SCHEMA, P.ROUTINE_NAME, C.id
FROM INFORMATION_SCHEMA.ROUTINES AS P
LEFT JOIN syscomments AS C ON
        OBJECT_ID('[' + P.ROUTINE_SCHEMA + '].[' + P.ROUTINE_NAME + ']') = C.id
        AND C.encrypted = 1
WHERE P.ROUTINE_TYPE = 'PROCEDURE'
AND C.id IS NOT NULL;

OPEN ProcedureCursor;

FETCH NEXT FROM ProcedureCursor
INTO @ObjectOwnerOrSchema, @ObjectName, @ObjectID;

WHILE @@FETCH_STATUS = 0
BEGIN
        -- Check that the provided object exists in the database.
        IF @ObjectID IS NULL
        BEGIN
                RAISERROR('The object name or schema provided does not exist in the database', 16, 1)
                RETURN
        END

        -- Check that the provided object is encrypted.
        IF NOT EXISTS(SELECT TOP 1 * FROM syscomments WHERE id = @ObjectID AND encrypted = 1)
        BEGIN
                RAISERROR('The object provided exists however it is not encrypted. Aborting.', 16, 1)
                RETURN
        END

        -- Determine the type of the object
        IF OBJECT_ID('[' + @ObjectOwnerOrSchema + '].[' + @ObjectName + ']', 'PROCEDURE') IS NOT NULL
                SET @ObjectType = 'PROCEDURE'
        ELSE
                IF OBJECT_ID('[' + @ObjectOwnerOrSchema + '].[' + @ObjectName + ']', 'TRIGGER') IS NOT NULL
                        SET @ObjectType = 'TRIGGER'
                ELSE
                        IF OBJECT_ID('[' + @ObjectOwnerOrSchema + '].[' + @ObjectName + ']', 'VIEW') IS NOT NULL
                                SET @ObjectType = 'VIEW'
                        ELSE
                                SET @ObjectType = 'FUNCTION'

        -- Get the binary representation of the object- syscomments no longer holds
        -- the content of encrypted object.
        SELECT TOP 1 @ContentOfEncryptedObject = imageval
        FROM sys.sysobjvalues
        WHERE objid = OBJECT_ID('[' + @ObjectOwnerOrSchema + '].[' + @ObjectName + ']')
                        AND valclass = 1 and subobjid = 1

        SET @ObjectDataLength = DATALENGTH(@ContentOfEncryptedObject)/2

        -- We need to alter the existing object and make it into a dummy object
        -- in order to decrypt its content. This is done in a transaction
        -- (which is later rolled back) to ensure that all changes have a minimal
        -- impact on the database.
        SET @ContentOfFakeObject = N'ALTER ' + @ObjectType + N' [' + @ObjectOwnerOrSchema + N'].[' +
@ObjectName + N'] WITH ENCRYPTION AS'

        WHILE DATALENGTH(@ContentOfFakeObject)/2 < @ObjectDataLength
        BEGIN
                        IF DATALENGTH(@ContentOfFakeObject)/2 + 4000 < @ObjectDataLength
                                        SET @ContentOfFakeObject = @ContentOfFakeObject + REPLICATE(N'-', 4000)
                        ELSE
                                        SET @ContentOfFakeObject = @ContentOfFakeObject + REPLICATE(N'-',
@ObjectDataLength - (DATALENGTH(@ContentOfFakeObject)/2))
        END

        -- Since we need to alter the object in order to decrypt it, this is done
        -- in a transaction
        SET XACT_ABORT OFF
```

```
        BEGIN TRAN

        EXEC(@ContentOfFakeObject)

        IF @@ERROR <> 0
                        ROLLBACK TRAN

        -- Get the encrypted content of the new "fake" object.
        SELECT TOP 1 @ContentOfFakeEncryptedObject = imageval
        FROM sys.sysobjvalues
        WHERE objid = OBJECT_ID('[' + @ObjectOwnerOrSchema + '].[' + @ObjectName + ']')
                        AND valclass = 1 and subobjid = 1

        IF @@TRANCOUNT > 0
                        ROLLBACK TRAN

        -- Generate a CREATE script for the dummy object text.
        SET @ContentOfFakeObject = N'CREATE ' + @ObjectType + N' [' + @ObjectOwnerOrSchema + N'].[' +
@ObjectName + N'] WITH ENCRYPTION AS'

        WHILE DATALENGTH(@ContentOfFakeObject)/2 < @ObjectDataLength
        BEGIN
                        IF DATALENGTH(@ContentOfFakeObject)/2 + 4000 < @ObjectDataLength
                                        SET @ContentOfFakeObject = @ContentOfFakeObject + REPLICATE(N'-', 4000)
                        ELSE
                                        SET @ContentOfFakeObject = @ContentOfFakeObject + REPLICATE(N'-',
@ObjectDataLength - (DATALENGTH(@ContentOfFakeObject)/2))
        END

        SET @i = 1

        --Fill the variable that holds the decrypted data with a filler character
        SET @ContentOfDecryptedObject = N''

        WHILE DATALENGTH(@ContentOfDecryptedObject)/2 < @ObjectDataLength
        BEGIN
                        IF DATALENGTH(@ContentOfDecryptedObject)/2 + 4000 < @ObjectDataLength
                                        SET @ContentOfDecryptedObject = @ContentOfDecryptedObject + REPLICATE
(N'A', 4000)
                        ELSE
                                        SET @ContentOfDecryptedObject = @ContentOfDecryptedObject + REPLICATE
(N'A', @ObjectDataLength - (DATALENGTH(@ContentOfDecryptedObject)/2))
        END


        WHILE @i <= @ObjectDataLength
        BEGIN
                        --xor real & fake & fake encrypted
                        SET @ContentOfDecryptedObject = STUFF(@ContentOfDecryptedObject, @i, 1,
                                        NCHAR(
                                                        UNICODE(SUBSTRING(@ContentOfEncryptedObject, @i, 1)) ^
                                                        (
                                                                        UNICODE(SUBSTRING(@ContentOfFakeObject,
@i, 1)) ^
                                                                        UNICODE(SUBSTRING
(@ContentOfFakeEncryptedObject, @i, 1))
                                                        )))

                        SET @i = @i + 1
        END

        SET @ContentOfDecryptedObject = REPLACE(REPLACE(@ContentOfDecryptedObject, 'With Encryption', ''),
'Create Procedure', 'Alter Procedure');

        EXECUTE (@ContentOfDecryptedObject)

        FETCH NEXT FROM ProcedureCursor
        INTO @ObjectOwnerOrSchema, @ObjectName, @ObjectID;
END

CLOSE ProcedureCursor;
```

```
DEALLOCATE ProcedureCursor;
```

Verwandte Artikel

- SQL Server decrypt stored procedures
- SQL Server rebuild all indexes
- SQL Server report index fragmentation
- Kill SQL-Server Database connections