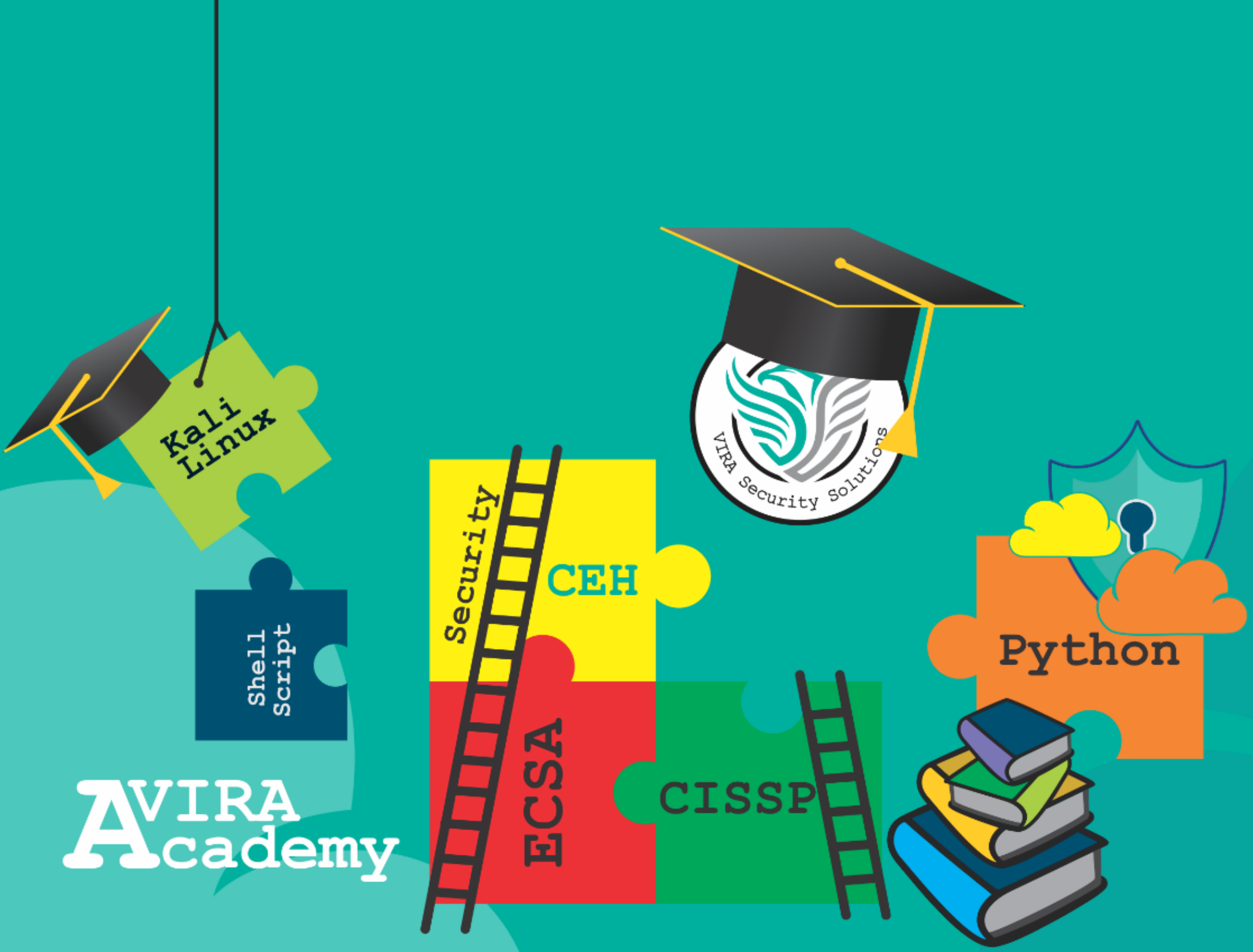
VIRA Security Solutions

راهکارهای امن ویرا

شماره ثبت: ۵۴۰۱۳۹

VIRA Security Solutions


Kali Linux
Shell Script
Security
CEH
ECSA
CISSP
Python
VIRA Security Solutions
VIRA Academy

DBA

SQL Server Administration

Part5

Mohammad Reza Gerami
mrgerami@aut.ac.ir
gerami@virasec.ir

Attach and detach a database

Attach and detach a database

Select DB
Single User Mode
Detach
Task -> Detach

And then you can attached again

```
EXEC sp_change_users_login 'Update_One','vira','vira'
```

Database Backup Strategies

Database Backup Strategies

Two leading questions direct the restore strategy:

What is the maximum amount of data can be tolerated?
Involves taking multiple transaction logs backup.

What is the accepted time frame for the database to be restored in case of a disaster?
Involves the use of high availability solutions.

To answer these questions you must ask the **why, when, where an who** for the strategy.

Database Backup Strategies

Why Back-up?

Bakups are taken in order to be able to restore the database to its original state just before the disaster and protect against the following reasons:

- ❖ Hardware failure
- ❖ Malicious damage
- ❖ User-error
- ❖ Database corruption

Database Backup Strategies

What Back-up?

Simply put, it’s an image of the database at the time of the full backup.

Database Backup Strategies

A **full database backup** backs up the whole database. It includes some part of the transactional log so that you could restore your database to the point when the full backup was finished. Usually, files with full database backup have '.bak' extension. It is recommended to periodically create full backups, but since it contains transaction log along with whole database data it takes significant space. To create full backup use the following command:

```
BACKUP DATABASE AdventureWorks TO DISK = 'C:\AdventureWorks.BAK' GO
```

Database Backup Strategies

If your database is big enough, it may become quite space-consuming to create full backups each time. Here a differential backup comes to the rescue. This kind of database backup is related to the last full backup and contains all changes that have been made since the last full backup. Each next differential backup contains the same data that was stored in the previous one, but none of them contain transactional logs. You need a previous full database backup to restore a differential backup. The file extension of a differential backup is usually ‘.dif’.

```
BACKUP DATABASE Adventureworks TO DISK = 'adventureworks.dif' WITH DIFFERENTIAL
```

For relatively small and rarely changing databases a full backup is often sufficient. Otherwise, you need to look at other backup types.

Database Backup Strategies

If it is crucial to restore your database close to the point of a failure or at any other point in time then you need to consider transactional log backups. This SQL Server backup type is possible only with full or bulk-logged recovery models. A transaction log backup contains all log records that have not been included in the last transaction log backup (or the last full backup). To create a transaction log backup to your database use the following command:

```
BACKUP LOG Adventureworks TO DISK = 'adventureworks.trn'
```

Database Backup Strategies

Use COPY_ONLY option if you need to make an additional full or transaction log backups which will occur beyond the regular sequence of SQL Server backups. To perform copy-only backup simply add “COPY_ONLY” clause:

```
BACKUP DATABASE Adventureworks TO DISK = 'full.bak' WITH COPY_ONLY
```

Database Backup Strategies

File and Filegroup Backups

These backup types allow you to backup one or more database files or filegroups. To execute file backup use the following command:

```
BACKUP DATABASE Adventureworks
FILE = 'File'
TO DISK = 'File.bck‘
```

Use this command to perform filegroup backup:

```
BACKUP DATABASE Adventureworks
FILEGROUP = 'Group'
TO DISK = 'Group.bck'
```

Database Backup Strategies

Partial Database Backup

Typically partial backups are used in simple recovery model to make backups of very large databases that have one or more read-only filegroups. However, SQL Server also allows making partial backups with full or bulk-logged recovery models. Use the following T-SQL command to create a partial backup:

```
BACKUP DATABASE Adventureworks READ_WRITE_FILEGROUPS TO
DISK = 'partial_backup.bak'
```

Database Backup Strategies

Overwrite or append backup sets

Another option is the WITH INIT and WITH NOINIT options. By default, the NOINIT option is enabled. It means that the backup will append to other backups in the file. For example, if you already have a full and a differential backup in full.bak, they will all remain after executing:

```
BACKUP DATABASE Adventureworks TO DISK = 'c:\backup\full.bak' WITH NOINIT
```

On the other hand, if you want to overwrite existing backups, the WITH INIT option will erase the previous set of backups:

```
BACKUP DATABASE Adventureworks TO DISK = 'c:\backup\full.bak' WITH INIT
```

Database Backup Strategies

Set expiration dates for backups

If you want your backup to expire, you can use the WITH EXPIREDATE option. The following example shows how to back up with an expiration date on March 28, 2018:

```
BACKUP DATABASE Adventureworks TO DISK = 'c:\backup\full.bak' WITH EXPIREDATE = N'03/28/2018 00:00:00'T
```

Another option is the option to expire after a specified number of days. The following example, shows how to retain a backup for 3 days:

```
BACKUP DATABASE Adventureworks TO DISK = 'c:\backup\full.bak' WITH RETAINDAYS = 3
```

Database Backup Strategies

Encrypt a database backup

Another interesting option is the backup with encryption, this option will allow encrypting our backup. To do that, we will need to create a master key:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '2455678KL95234nl0zBe'
```

Next, we will need to create a certificate:

```
CREATE CERTIFICATE MyServerCert WITH SUBJECT = 'my certificate'
```

Database Backup Strategies

Finally, we can backup our database:

```
BACKUP DATABASE [testdb] TO DISK = 'c:\backup\full.bak'
WITH FORMAT,
ENCRYPTION
(
 ALGORITHM = AES_256,
 SERVER CERTIFICATE = MyServerCert ),
 STATS = 10
GO
```

Database Backup Strategies

Backup compression

In addition, it is possible to compress your backup using the option WITH COMPRESSION. This option will compress your backup:

```
BACKUP DATABASE Adventureworks TO DISK = 'c:\backup\full.bak' WITH FORMAT, COMPRESSION
```

Note that the backup compression is only supported on SQL Server Enterprise and Standard editions (and the Business Intelligence Edition in SQL Server 2012).

Database Backup Strategies

SQL Server Backup Tips

If your database is big, you will need to combine full, differential and transaction log backups. If your database is big and it does not change too much, a differential backup will take less space than a full backup and you will save a lot of space.
Do not store your backup on the same drive than the database. If possible try to store your backup on another Server or even better in another physical place. If your hard drive fails and you have the database and backups, you may not be able to recover your data.
Test your backups to make sure that they are working fine. There are some nice options that can be useful to verify that the backup is OK, like the verifyonly option. The following example will create a full backup and then test if it works using the RESTORE WITH VERIFYONLY option to just verify if the backup is OK:

Database Backup Strategies

SQL Server Backup Tips

```
BACKUP DATABASE Adventureworks TO DISK = 'c:\backup\full.bak'
GO

declare @backupSetId as int

select @backupSetId = position from msdb..backupset
where database_name=N'adventureworks' and backup_set_id=(select
max(backup_set_id)
from msdb..backupset
where database_name=N'adventureworks' )

if @backupSetId is null
begin raiserror (N'Verify failed. Backup information for database
''adventureworks'' not found.' , 16, 1) end

RESTORE VERIFYONLY FROM DISK = 'c:\backup\full.bak' WITH FILE = @backupsetid
```

Recovering Data from the SQL Server Transaction Log

Recovering Data from the SQL Server Transaction Log

The SQL Server Transaction Log plays also an important role in recovering deleted or modified data if you mistakenly perform a DELETE or UPDATE operation with the wrong condition, or badly without filtration condition. This can be achieved by listening to the records stored inside in this black box that is called SQL Server Transaction Log file. The events that are written to the SQL Server Transaction Log file, without any additional configuration required from the database administrator side, includes the different types of DML operations, such as INSERT, UPDATE and DELETE statements, and DDL operations, such as CREATE and DROP statements.

Recovering Data from the SQL Server Transaction Log

To compare the different methods that we can use to recover data from the SQL Server Transaction Log file, we will create a new database with a single table, using the script below:

```
CREATE DATABASE RecoverFromTransactionLog
GO
USE RecoverFromTransactionLog
GO
CREATE TABLE [dbo].[Employee_Main](
  [Emp_ID] [int] IDENTITY(1,1) NOT NULL,
  [EMP_FirsrName] [varchar](50) NULL,
  [EMP_LastName] [varchar](50) NULL,
  [EMP_BirthDate] [datetime] NULL,
  [EMP_PhoneNumber] [varchar](50) NULL,
  [EMP_Address] [varchar](max) NULL,
PRIMARY KEY CLUSTERED
(
  [Emp_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_
LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Recovering Data from the SQL Server Transaction Log

After creating the database and the table, we will fill that table with 30 records, using the INSERT INTO T-SQL statements below:

```
USE [RecoverFromTransactionLog]
GO
INSERT INTO [dbo].[Employee_Main] VALUES ('AAA','BBB','1980-10-15','98542224','ABCDE')
GO 10
INSERT INTO [dbo].[Employee_Main] VALUES ('CCC','DDD','1985-08-07','98543334','FGHIJ')
GO 10
INSERT INTO [dbo].[Employee_Main] VALUES ('EEE','FFF','1989-04-06','98544454','KLMNO')
GO 10
```

Recovering Data from the SQL Server Transaction Log

Having no backup taken from the database yet, how could we recover the table’s data if the below UPDATE statement is executed over that table with no WHERE clause used to filter the data to be modified?

```
UPDATE [dbo].[Employee_Main] SET [EMP_PhoneNumber]='98544455'
```

Recovering Data from the SQL Server Transaction Log

Using Built-in Methods

fn_dblog
fn_dump_dblog()
DBCC Page Command

3rd party solution

ApexSQL tools

Recovering Data from the SQL Server Transaction Log

fn_dblog

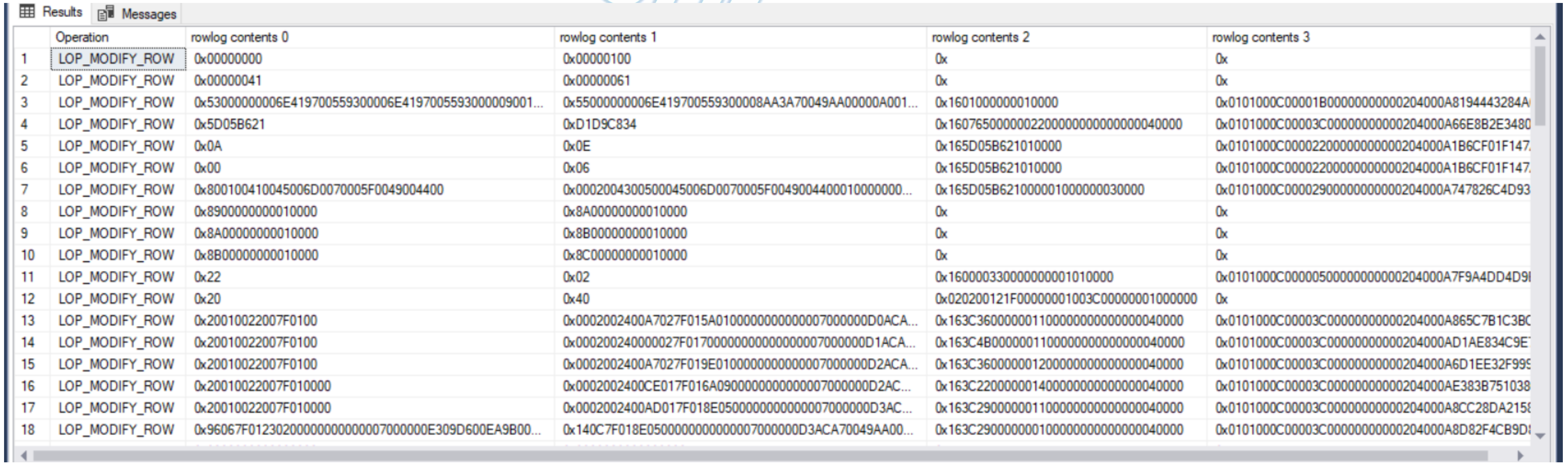
The content of the SQL Server Transaction Log can be read online using a number of built-in functions. The first function that we will try to use to recover the lost data is the **fn_dblog**. It is an undocumented system table-valued function, used to dive in the SQL Server Transaction Log and view the content of the active portion of that file.
In order to use the fn_dblog() function, you need to provide two parameters, the start and the end LSN of the log. To view all logs available in the active portion of the SQL Transaction Log file, you can pass NULL value for these two parameters. As the result returned from the fn_dblog() function contains 129 columns, it is better to choose only the columns that contain the data you are interested in and filter the result to show only the operation that you plan to recover. For example, the LOP_INSERT_ROWS operation is logged when a new record is inserted, LOP_DELETE_ROWS operation is logged when an existing record is deleted, and LOP_MODIFY_ROW operation is logged when an existing record is updated. The below script can be used to check the update statements that are performed on that database:

Recovering Data from the SQL Server Transaction Log

fn_dblog

```
SELECT Operation,
      [rowlog contents 0],
    [rowlog contents 1],
    [rowlog contents 2],
    [rowlog contents 3]
FROM sys.fn_dblog(NULL,NULL)
WHERE operation = ('LOP_MODIFY_ROW')
ORDER BY Operation ASC
```

And the result in our example will be like:



	Operation	rowlog contents 0	rowlog contents 1	rowlog contents 2	rowlog contents 3
1	LOP_MODIFY_ROW	0x00000000	0x00000100	0x	0x
2	LOP_MODIFY_ROW	0x00000041	0x00000061	0x	0x
3	LOP_MODIFY_ROW	0x53000000006E419700559300006E4197005593000009001...	0x55000000006E419700559300008AA3A70049AA00000A001...	0x160100000010000	0x0101000C00001B0000000000204000A8194443284A
4	LOP_MODIFY_ROW	0x5D05B621	0xD1D9C834	0x16076500000022000000000000000040000	0x0101000C00003C0000000000204000A66E8B2E3480
5	LOP_MODIFY_ROW	0x0A	0x0E	0x165D05B621010000	0x0101000C00002200000000000204000A1B6CF01F147
6	LOP_MODIFY_ROW	0x00	0x06	0x165D05B621010000	0x0101000C00002200000000000204000A1B6CF01F147
7	LOP_MODIFY_ROW	0x800100410045006D0070005F0049004400	0x000200430050045006D0070005F0049004400010000000...	0x165D05B62100000100000030000	0x0101000C00002900000000000204000A747826C4D93
8	LOP_MODIFY_ROW	0x8900000000010000	0x8A00000000010000	0x	0x
9	LOP_MODIFY_ROW	0x8A00000000010000	0x8B00000000010000	0x	0x
10	LOP_MODIFY_ROW	0x8B00000000010000	0x8C00000000010000	0x	0x
11	LOP_MODIFY_ROW	0x22	0x02	0x16000033000000001010000	0x0101000C00000500000000000204000A7F9A4DD4D9
12	LOP_MODIFY_ROW	0x20	0x40	0x020200121F0000001003C0000001000000	0x
13	LOP_MODIFY_ROW	0x20010022007F0100	0x0002002400A7027F015A01000000000000007000000D0ACA...	0x163C36000000110000000000000040000	0x0101000C00003C00000000000204000A865C7B1C3B
14	LOP_MODIFY_ROW	0x20010022007F0100	0x000200240000027F01700000000000000007000000D1ACA...	0x163C4B000000110000000000000040000	0x0101000C00003C00000000000204000AD1AE834C9E
15	LOP_MODIFY_ROW	0x20010022007F0100	0x0002002400A7027F019E01000000000000007000000D2ACA...	0x163C36000000120000000000000040000	0x0101000C00003C00000000000204000A6D1EE32F99
16	LOP_MODIFY_ROW	0x20010022007F010000	0x0002002400CE017F016A09000000000000007000000D2AC...	0x163C22000000140000000000000040000	0x0101000C00003C00000000000204000AE383B751038
17	LOP_MODIFY_ROW	0x20010022007F010000	0x0002002400AD017F018E05000000000000007000000D3AC...	0x163C29000000110000000000000040000	0x0101000C00003C00000000000204000A8CC28DA215
18	LOP_MODIFY_ROW	0x96067F01230200000000000000007000000E309D600EA9B00...	0x140C7F018E05000000000000007000000D3ACA70049AA00...	0x163C29000000100000000000000040000	0x0101000C00003C00000000000204000A8D82F4CB9D

Recovering Data from the SQL Server Transaction Log

fn_dblog

You can see from the previous result, that recovering the data from the Transaction Log file using the fn_dblog() function is complex, as the data is viewed in hexadecimal format and spread in multiple columns.

In addition, the UPDATE operation will be minimally logged, with no information about the new or old value, as it writes only the part that is changed, for example, changing the number 4 to 5 in the phone number as performed previously. In order to be able to recover the whole column value, you need to reconstruct it manually using the correct data type, by linking it with the insert operation performed prior to updating the value.

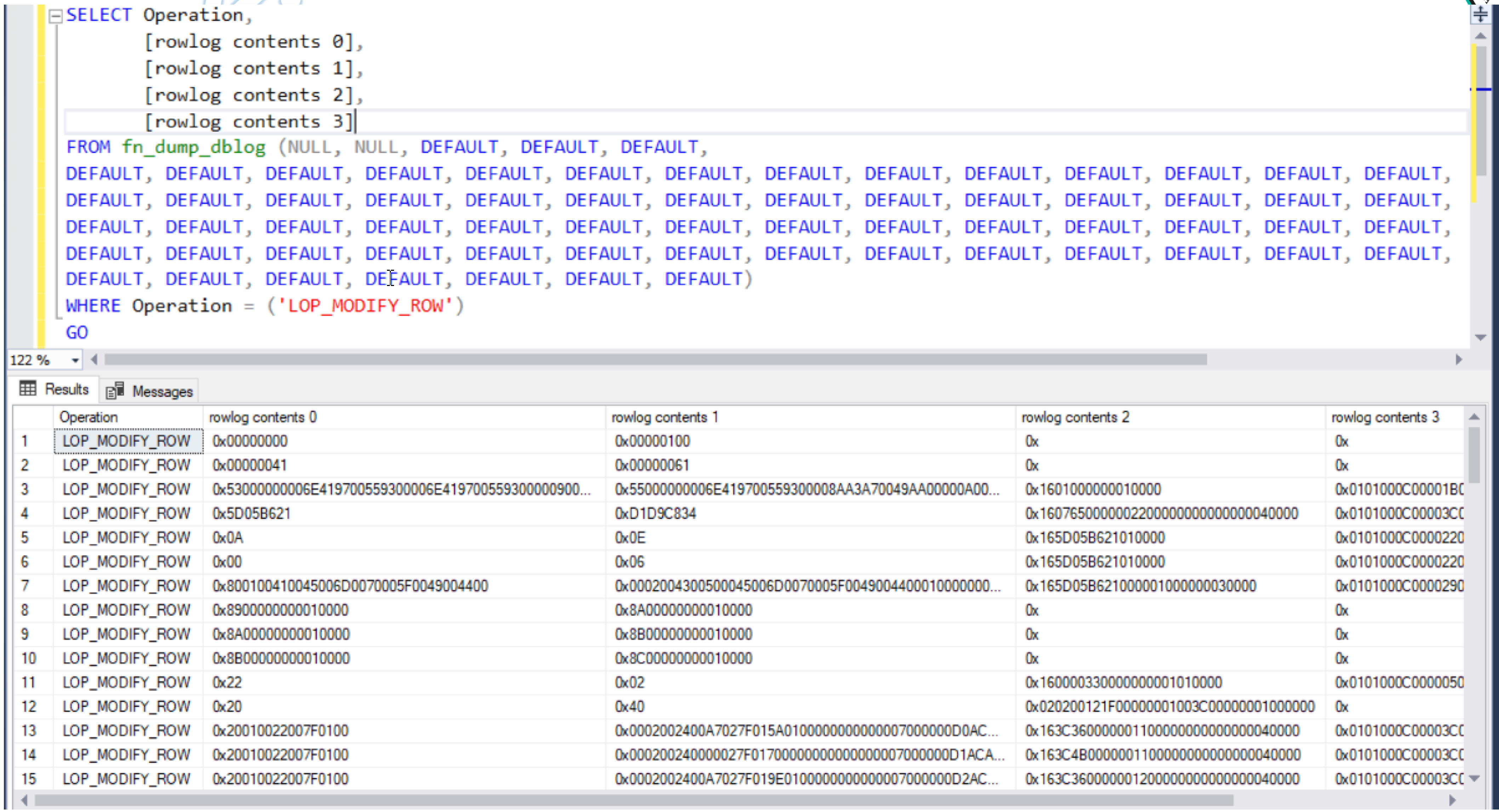
Recovering Data from the SQL Server Transaction Log

fn_dump_dblog()

The second system built-in function, **fn_dump_dblog()**, has one advantage over the fn_dblog() function is that it can be used to read the SQL Server Transaction Log backup, in addition to the ability to read the online Transaction Log file. On the other hand, in order to use the fn_dump_dblog() function, you need to provide 68 mandatory parameters, as shown in the script below:

Recovering Data from the SQL Server Transaction Log

fn_dump_dblog()



```
SELECT Operation,
       [rowlog contents 0],
       [rowlog contents 1],
       [rowlog contents 2],
       [rowlog contents 3]
FROM fn_dump_dblog (NULL, NULL, DEFAULT, DEFAULT, DEFAULT,
DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT,
DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT,
DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT,
DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT,
DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT)
WHERE Operation = ('LOP_MODIFY_ROW')
GO
```

122 %

	Operation	rowlog contents 0	rowlog contents 1	rowlog contents 2	rowlog contents 3
1	LOP_MODIFY_ROW	0x00000000	0x00000100	0x	0x
2	LOP_MODIFY_ROW	0x00000041	0x00000061	0x	0x
3	LOP_MODIFY_ROW	0x53000000006E419700559300006E419700559300000900...	0x5500000006E419700559300008AA3A70049AA00000A00...	0x160100000010000	0x0101000C00001B
4	LOP_MODIFY_ROW	0x5D05B621	0xD1D9C834	0x16076500000022000000000000000040000	0x0101000C00003C
5	LOP_MODIFY_ROW	0x0A	0x0E	0x165D05B621010000	0x0101000C000022
6	LOP_MODIFY_ROW	0x00	0x06	0x165D05B621010000	0x0101000C000022
7	LOP_MODIFY_ROW	0x800100410045006D0070005F0049004400	0x0002004300500045006D0070005F004900440001000000...	0x165D05B62100000100000000030000	0x0101000C000029
8	LOP_MODIFY_ROW	0x8900000000010000	0x8A00000000010000	0x	0x
9	LOP_MODIFY_ROW	0x8A00000000010000	0x8B00000000010000	0x	0x
10	LOP_MODIFY_ROW	0x8B00000000010000	0x8C00000000010000	0x	0x
11	LOP_MODIFY_ROW	0x22	0x02	0x16000033000000000001010000	0x0101000C000005
12	LOP_MODIFY_ROW	0x20	0x40	0x020200121F0000001003C0000001000000	0x
13	LOP_MODIFY_ROW	0x20010022007F0100	0x0002002400A7027F015A010000000000000007000000D0AC...	0x163C360000001100000000000000040000	0x0101000C00003C
14	LOP_MODIFY_ROW	0x20010022007F0100	0x000200240000027F017000000000000000007000000D1ACA...	0x163C4B000000110000000000000040000	0x0101000C00003C
15	LOP_MODIFY_ROW	0x20010022007F0100	0x0002002400A7027F019E010000000000000007000000D2AC...	0x163C360000001200000000000000040000	0x0101000C00003C

Recovering Data from the SQL Server Transaction Log

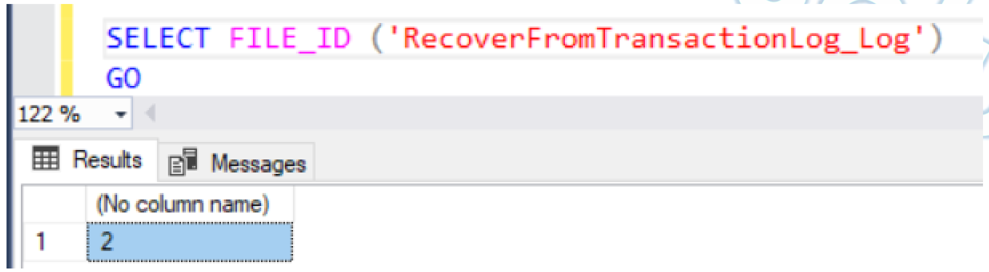
fn_dump_dblog()

You can see from the previous snapshot that the fn_dump_dblog() function returns the same result as the result returned from the fn_dblog() function, with the same complexity in reading the hexadecimal data and reconstructing it manually to get useful information.

Recovering Data from the SQL Server Transaction Log

DBCC Page Command

SQL Server provides us with a large number of DBCC commands that can be used to perform the different types of SQL Server administration tasks. One of these undocumented commands is the DBCC Page command, that can be used to read the content of the database data and log online files. In order to view the content of the database log file, you need to provide the database name and the ID of the Transaction Log file. The ID of the Transaction Log file can be retrieved from the query below:



```
SELECT FILE_ID ('RecoverFromTransactionLog_Log')
GO
```

	(No column name)
1	2

Recovering Data from the SQL Server Transaction Log

DBCC Page Command

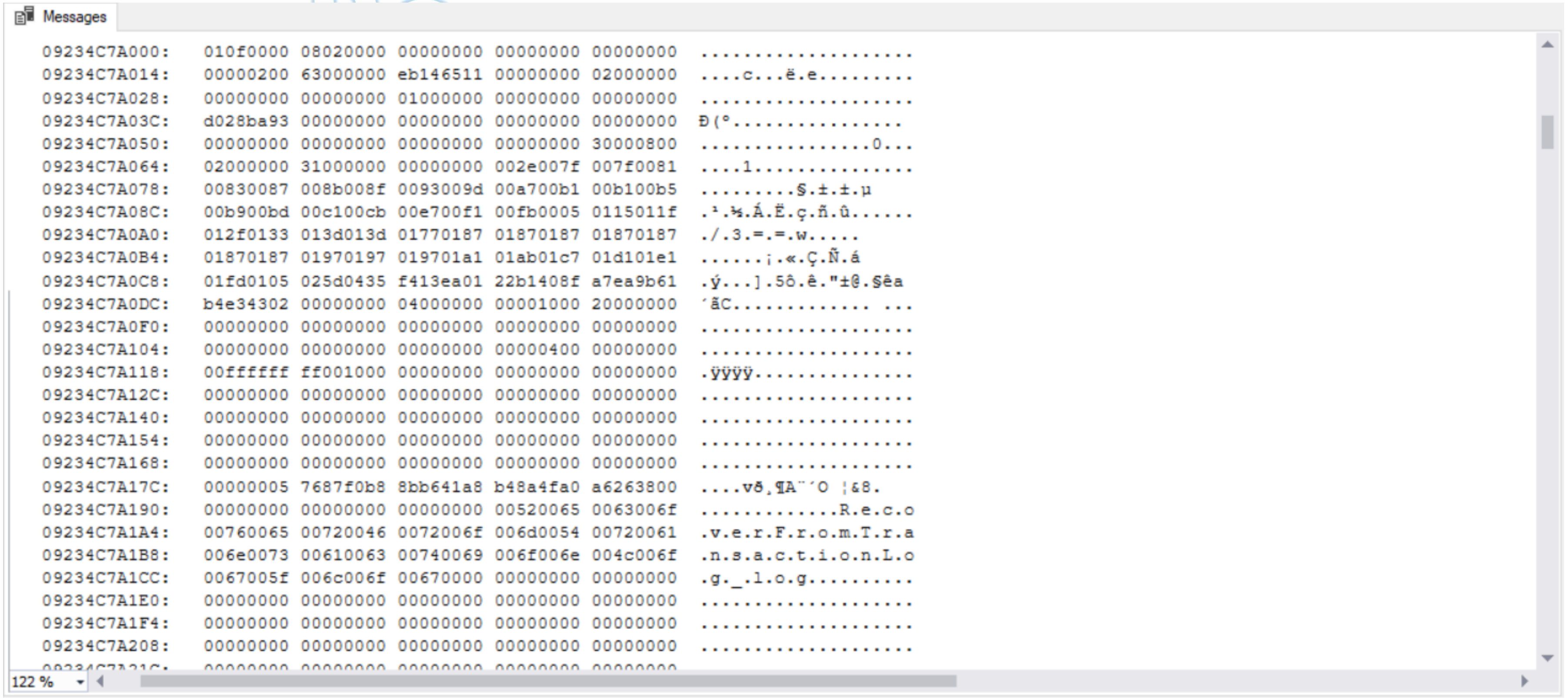
Then you will be able to view the Transaction Log file, after turning on the Trace Flag 3604, in order to print the output of the DBCC command, as shown below:

```
DBCC TRACEON (3604, -1)
GO
DBCC PAGE (RecoverFromTransactionLog, 2, 0, 2)
GO
```

Recovering Data from the SQL Server Transaction Log

DBCC Page Command

Again, the result returned from the DBCC command will be displayed in hexadecimal format, that needs big effort translating it to a meaningful shape and using the correct data type, as below:



Messages

```
09234C7A000:   010f0000 08020000 00000000 00000000 00000000  ....................
09234C7A014:   00000200 63000000 eb146511 00000000 02000000  ....c...ë.e.........
09234C7A028:   00000000 00000000 01000000 00000000 00000000  ....................
09234C7A03C:   d028ba93 00000000 00000000 00000000 00000000  Ð(°.................
09234C7A050:   00000000 00000000 00000000 00000000 30000800  ................0...
09234C7A064:   02000000 31000000 00000000 002e007f 007f0081  ....1...............
09234C7A078:   00830087 008b008f 0093009d 00a700b1 00b100b5  .........§.±.±.µ
09234C7A08C:   00b900bd 00c100cb 00e700f1 00fb0005 0115011f  .¹.½.Á.Ë.ç.ñ.û......
09234C7A0A0:   012f0133 013d013d 01770187 01870187 01870187  ./.3.=.=.w.....
09234C7A0B4:   01870187 01970197 019701a1 01ab01c7 01d101e1  ......¡.«.Ç.Ñ.á
09234C7A0C8:   01fd0105 025d0435 f413ea01 22b1408f a7ea9b61  .ý...].5ô.ê."±@.§êa
09234C7A0DC:   b4e34302 00000000 04000000 00001000 20000000  ´ãC............ ...
09234C7A0F0:   00000000 00000000 00000000 00000000 00000000  ....................
09234C7A104:   00000000 00000000 00000000 00000400 00000000  ....................
09234C7A118:   00ffffff ff001000 00000000 00000000 00000000  .ÿÿÿÿ...............
09234C7A12C:   00000000 00000000 00000000 00000000 00000000  ....................
09234C7A140:   00000000 00000000 00000000 00000000 00000000  ....................
09234C7A154:   00000000 00000000 00000000 00000000 00000000  ....................
09234C7A168:   00000000 00000000 00000000 00000000 00000000  ....................
09234C7A17C:   00000005 7687f0b8 8bb641a8 b48a4fa0 a6263800  ....vð¸¶A¨´O ¦&8.
09234C7A190:   00000000 00000000 00000000 00520065 0063006f  .............R.e.c.o
09234C7A1A4:   00760065 00720046 0072006f 006d0054 00720061  .v.e.r.F.r.o.m.T.r.a
09234C7A1B8:   006e0073 00610063 00740069 006f006e 004c006f  .n.s.a.c.t.i.o.n.L.o
09234C7A1CC:   0067005f 006c006f 00670000 00000000 00000000  .g._.l.o.g..........
09234C7A1E0:   00000000 00000000 00000000 00000000 00000000  ....................
09234C7A1F4:   00000000 00000000 00000000 00000000 00000000  ....................
09234C7A208:   00000000 00000000 00000000 00000000 00000000  ....................
09234C7A21C:   00000000 00000000 00000000 00000000 00000000
```

122 %

Recovering Data from the SQL Server Transaction Log

3rd party solution

When a database is corrupted or table's data is mistakenly lost or modified, the long time that is required to reconstruct and parse the data using the built-in methods cannot be acceptable, as the seconds of downtime for the international companies means losing potentially a lot of money. So, we need search in the SQL Server administration market for a tool that can be easily and quickly used to read the Transaction Log file content and show this content in a user-friendly format.

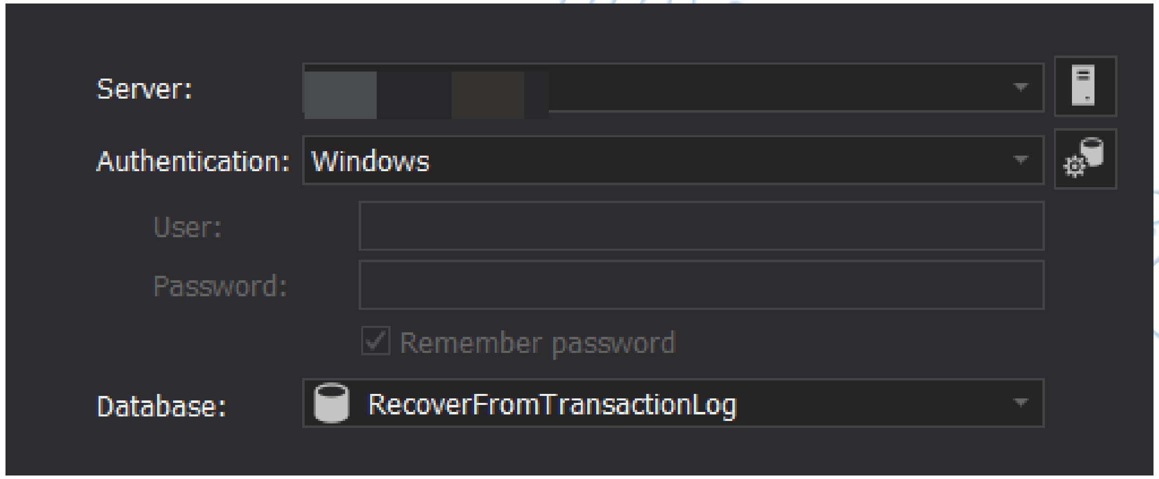
ApexSQL Log is a SQL Server Transaction Log reader tool, that can be easily used to read the content of the online SQL Transaction Log file, the detached Transaction Log file or Transaction Log backup files chain. You need only to go through a simple wizard, that contains a variant number of customized options, and ApexSQL Log will perform the reconstruction, translation, and parsing automatically, without the need to perform any manual task.

Before using ApexSQL Log, you need to download it from the ApexSQL tools download page, then proceed with the installation process, by following a straight-forward installation wizard. After installing ApexSQL Log, you can run that tool by clicking on the Axe icon.

Recovering Data from the SQL Server Transaction Log

3rd party solution

In the start page, provide the connection information, includes the SQL Server instance name, the authentication mode, the credentials, and the database name, that will be used to connect to your SQL Server instance and read the database Transaction Log File content, as shown below:



Server:	[redacted]
Authentication:	Windows
User:	
Password:	
	✓ Remember password
Database:	RecoverFromTransactionLog