<http://www.sqler.com/325804#2>

고희수

트랜잭션 로그 읽기 입니다.
추적시 범위를 좁히며 접근할 수 있어 유용(?) 할 것 같네요.

CREATE DATABASE [Crack\_Me];

GO

USE Crack\_Me;

GO

/\* 테스트테이블 생성및 데이터를 한개 입력합니다. \*/

CREATE TABLE [dbo].[Crack\_Me\_If\_You\_Can](

 [ID] [int] PRIMARY KEY IDENTITY NOT NULL,

 [Insert\_Date] [datetime] NOT NULL,

 [Some\_Data] [varchar](100) NOT NULL,

 [Optional\_Data] [varchar](50)NULL,

 [Life\_the\_Universe\_and\_Everything] [int] NOT NULL,

);

GO

INSERT INTO [Crack\_Me\_If\_You\_Can]

(

 Insert\_Date,

 Some\_Data,

 Optional\_Data,

 Life\_the\_Universe\_and\_Everything

)

VALUES (GetDate(), 'Don''t Panic', 'Share and Enjoy', 42)

GO

SELECT \* FROM Crack\_Me\_If\_You\_Can;

GO

/\* fn\_dblog함수로 트랜잭션로그 내용을 확인 할 수 있습니다. \*/

SELECT \* FROM fn\_dblog(NULL, NULL)

GO

/\* 개체별allocation\_unit\_id를 확인 할 수 있습니다.

sys.allocation\_units 의

type이1 또는3이면container\_id는sys.partitions.hobt\_id 입니다.

type이2이면container\_id = sys.partitions.partition\_id 입니다.

\*/

SELECT

 allocunits.allocation\_unit\_id

 ,objects.name

 ,objects.id

FROM sys.allocation\_units allocunits

INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)

 AND partitions.hobt\_id = allocunits.container\_id)

 OR (allocunits.type = 2 and partitions.partition\_id = allocunits.container\_id)

INNER JOIN sysobjects objects ON partitions.object\_id = objects.id

AND objects.type IN ('U', 'u')

WHERE

 partitions.index\_id IN (0, 1)

GO

SELECT

\*

FROM fn\_dblog(NULL, NULL)

WHERE

 AllocUnitId = 2105058535

AND Operation = 'LOP\_INSERT\_ROWS'

GO

/\* 활성로그뿐 아니라 전체로그를 보여주기 위해 추적플래그2537을 설정 합니다.

 아래스크립트는 입력된 데이터로그정보를 확인 하기 위함입니다.

fn\_dblog 함수의Operation:

0)AFTER TOTAL SHRINK (O KB)

LOP\_BEGIN\_CKPT

LOP\_END\_CKPT

LOP\_FILE\_HDR\_MODIFY

1)FOR INSERT STATEMENT:

LOP\_BEGIN\_XACT

LOP\_INSERT\_ROWS

LOP\_COMMIT\_XACT

2)FOR UPDATE STATEMENT:

LOP\_BEGIN\_XACT

LOP\_MODIFY\_ROW

LOP\_COMMIT\_XACT

3)FOR CREATE OBJECT STATEMENT:

LOP\_BEGIN\_XACT

LOP\_INSERT\_ROWS

LOP\_MARK\_DDL

\*/

DBCC TRACEON(2537)

SELECT

 [Current LSN],

 Operation,

 dblog.[Transaction ID],

 AllocUnitId,

 AllocUnitName,

 [Page ID],

 [Slot ID],

 [Num Elements],

 dblog1.[Begin Time],

 dblog1.[Transaction Name],

 [RowLog Contents 0],

 [Log Record]

FROM ::fn\_dblog(NULL, NULL) dblog

INNER JOIN (

 SELECT

 allocunits.allocation\_unit\_id

 ,objects.name

 ,objects.id

 FROM sys.allocation\_units allocunits

 INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)

 AND partitions.hobt\_id = allocunits.container\_id)

 OR (allocunits.type = 2 and partitions.partition\_id = allocunits.container\_id)

 INNER JOIN sysobjects objects ON partitions.object\_id = objects.id

 AND objects.type IN ('U', 'u')

 WHERE

 partitions.index\_id IN (0, 1)

) allocunits ON dblog.AllocUnitID = allocunits.allocation\_unit\_id

INNER JOIN (

 SELECT

 [Begin Time]

 ,[Transaction Name]

 ,[Transaction ID]

 FROM fn\_dblog(NULL, NULL) x

 WHERE Operation = 'LOP\_BEGIN\_XACT'

) dblog1 ON dblog1.[Transaction ID] = dblog.[Transaction ID]

WHERE

 [Page ID] IS NOT NULL AND [Slot ID] >= 0

AND dblog.[Transaction ID] != '0000:00000000'

AND Context in ('LCX\_HEAP', 'LCX\_CLUSTERED')

DBCC TRACEOFF(2537)

GO

TransactionLog\_1.jpg



/\*

페이지(PageID:0001:0000004d)내용을 테이블결과셋 형태로 확인 합니다.
\*/

DECLARE @pageID$ NVARCHAR(23), @pageID NVARCHAR(50), @sqlCmd NVARCHAR(4000);

SET @pageID$ = '0001:0000004d'

SELECT @pageID =

CONVERT(VARCHAR(4), CONVERT(INT, CONVERT(VARBINARY,

SUBSTRING(@pageID$, 0, 5), 2)))

+ ',' +

CONVERT(VARCHAR(8), CONVERT(INT, CONVERT(VARBINARY,

SUBSTRING(@pageID$, 6, 8), 2)))

SET @sqlCmd = 'DBCC PAGE (''Crack\_Me'',' + @pageID + ',3) WITH TABLERESULTS'

PRINT @sqlCmd

EXECUTE(@sqlCmd)

TransactionLog\_2.jpg



/\*

위 첫번째결과에서 나온 [RowLog Contents 0]의 값을 아래@RowLogContents 변수에 입력합니다.

위 두번째결과에서 나온 [Metadata: AllocUnitId]값 72057594039828480 을 아래 스크립트

(allocunits.allocation\_unit\_id = 72057594039697408) 입력합니다.

여기부터 끝까지 선택한 후 실행합니다.
\*/

DECLARE @RowLogContents VARBINARY(8000)

SET @RowLogContents = 0x300014000100000045040200449E00002A000000050000020028003700446F6E27742050616E6963536861726520616E6420456E6A6F79

DECLARE @lenFixedBytes SMALLINT, @noOfCols SMALLINT, @nullBitMapLength SMALLINT, @nullByteMap VARBINARY(MAX), @nullBitMap VARCHAR(MAX), @noVarCols SMALLINT, @columnOffsetArray VARBINARY(MAX), @varColPointer SMALLINT

SELECT

 @lenFixedBytes = CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING(@RowLogContents, 2 + 1, 2)))),

 @noOfCols = CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING(@RowLogContents, @lenFixedBytes + 1, 2)))),

 @nullBitMapLength = CONVERT(INT, ceiling(@noOfCols/8.0)),

 @nullByteMap = SUBSTRING(@RowLogContents, @lenFixedBytes + 3, @nullBitMapLength),

 @noVarCols = CASE WHEN SUBSTRING(@RowLogContents, 1, 1) = 0x30 THEN

 CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING(@RowLogContents, @lenFixedBytes + 3 + @nullBitMapLength, 2))))

 ELSE null

 END,

 @columnOffsetArray = CASE WHEN SUBSTRING(@RowLogContents, 1, 1) = 0x30 THEN

 SUBSTRING(@RowLogContents, @lenFixedBytes + 3 + @nullBitMapLength + 2, @noVarCols \* 2)

 ELSE null

 END,

 @varColPointer = CASE WHEN SUBSTRING(@RowLogContents, 1, 1) = 0x30 THEN

 (@lenFixedBytes + 2 + @nullBitMapLength + 2 + (@noVarCols \* 2))

 ELSE null

 END

DECLARE @byteTable TABLE

(

 byte INT

)

DECLARE @cnt INT

SET @cnt = 1

WHILE (@cnt < @nullBitMapLength + 1)

BEGIN

 INSERT INTO @byteTable(byte)

 VALUES(@cnt)

 SET @cnt = @cnt +1

END

SELECT

 @nullBitMap = COALESCE(@nullBitMap, '') +

 CONVERT(NVARCHAR(1), (SUBSTRING(@nullByteMap, byte, 1) / 128) % 2) +

 CONVERT(NVARCHAR(1), (SUBSTRING(@nullByteMap, byte, 1) / 64) % 2) +

 CONVERT(NVARCHAR(1), (SUBSTRING(@nullByteMap, byte, 1) / 32) % 2) +

 CONVERT(NVARCHAR(1), (SUBSTRING(@nullByteMap, byte, 1) / 16) % 2) +

 CONVERT(NVARCHAR(1), (SUBSTRING(@nullByteMap, byte, 1) / 8) % 2) +

 CONVERT(NVARCHAR(1), (SUBSTRING(@nullByteMap, byte, 1) / 4) % 2) +

 CONVERT(NVARCHAR(1), (SUBSTRING(@nullByteMap, byte, 1) / 2) % 2) +

 CONVERT(NVARCHAR(1), SUBSTRING(@nullByteMap, byte, 1) % 2)

FROM @byteTable b

ORDER BY byte DESC

SELECT

 SUBSTRING(@RowLogContents, 2 + 1, 2) AS lenFixedBytes,

 SUBSTRING(@RowLogContents, @lenFixedBytes + 1, 2) AS noOfCols,

 SUBSTRING(@RowLogContents, @lenFixedBytes + 3, @nullBitMapLength) AS nullByteMap,

 SUBSTRING(@RowLogContents, @lenFixedBytes + 3 + @nullBitMapLength, 2) AS noVarCols,

 SUBSTRING(@RowLogContents, @lenFixedBytes + 3 + @nullBitMapLength + 2, @noVarCols \* 2) AS columnOffsetArray,

 @lenFixedBytes + 2 + @nullBitMapLength + 2 + (@noVarCols \* 2) AS varColStart

SELECT

 @lenFixedBytes AS lenFixedBytes,

 @noOfCols AS noOfCols,

 @nullBitMapLength AS nullBitMapLength,

 @nullByteMap AS nullByteMap,

 @nullBitMap AS nullBitMap,

 @noVarCols AS noVarCols,

 @columnOffsetArray AS columnOffsetArray,

 @varColPointer AS varColStart

DECLARE @colOffsetTable TABLE

(

 colNum SMALLINT,

 columnOffset VARBINARY(2),

 columnOffvalue SMALLINT,

 columnLength SMALLINT

)

SET @cnt = 1

WHILE (@cnt <= @noVarCols)

BEGIN

 INSERT INTO @colOffsetTable(colNum, columnOffset, columnOffValue, columnLength)

 VALUES(

 @cnt \* - 1,

 SUBSTRING (@columnOffsetArray, (2 \* @cnt) - 1, 2),

 CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE (SUBSTRING (@columnOffsetArray, (2 \* @cnt) - 1, 2)))),

 CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE (SUBSTRING (@columnOffsetArray, (2 \* @cnt) - 1, 2))))

 - ISNULL(NULLIF(CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE (SUBSTRING (@columnOffsetArray, (2 \* (@cnt - 1)) - 1, 2)))), 0), @varColPointer)

 )

 SET @cnt = @cnt + 1

END

SELECT \* FROM @colOffsetTable

SELECT

 cols.leaf\_null\_bit AS nullbit,

 ISNULL(syscolumns.length, cols.max\_length) AS [length],

 CASE

 WHEN is\_uniqueifier = 1 THEN 'UNIQUIFIER'

 ELSE ISNULL(syscolumns.name, 'DROPPED')

 END [name],

 cols.system\_type\_id,

 cols.leaf\_bit\_position AS bitpos,

 ISNULL(syscolumns.xprec, cols.precision) AS xprec,

 ISNULL(syscolumns.xscale, cols.scale) AS xscale,

 cols.leaf\_offset,

 is\_uniqueifier

FROM sys.allocation\_units allocunits

INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)

 AND partitions.hobt\_id = allocunits.container\_id)

 OR(allocunits.type = 2 AND partitions.partition\_id =allocunits.container\_id)

INNER JOIN sys.system\_internals\_partition\_columns cols ON

 cols.partition\_id = partitions.partition\_id

LEFT OUTER JOIN syscolumns ON syscolumns.id = partitions.object\_id

 AND syscolumns.colid = cols.partition\_column\_id

WHERE allocunits.allocation\_unit\_id = 72057594039828480

ORDER BY nullbit

DECLARE @schema TABLE

(

 [column] INT,

 [length] INT,

 [name] NVARCHAR(255),

 [system\_type\_id] INT,

 [bitpos] INT,

 [xprec] INT,

 [xscale] INT,

 [leaf\_offset] INT,

 [is\_uniqueifier] BIT,

 [is\_null] BIT NULL

)

INSERT INTO @schema

SELECT

cols.leaf\_null\_bit AS nullbit,

ISNULL(syscolumns.length, cols.max\_length) AS [length],

CASE

WHEN is\_uniqueifier = 1 THEN 'UNIQUIFIER'

ELSE isnull(syscolumns.name, 'DROPPED')

END [name],

cols.system\_type\_id,

cols.leaf\_bit\_position AS bitpos,

ISNULL(syscolumns.xprec, cols.precision) AS xprec,

ISNULL(syscolumns.xscale, cols.scale) AS xscale,

cols.leaf\_offset,

is\_uniqueifier,

SUBSTRING(REVERSE(@nullBitMap), cols.leaf\_null\_bit, 1) AS is\_null

FROM sys.allocation\_units allocunits

INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)

 AND partitions.hobt\_id = allocunits.container\_id) OR (allocunits.type = 2 AND partitions.partition\_id = allocunits.container\_id)

INNER JOIN sys.system\_internals\_partition\_columns cols ON cols.partition\_id = partitions.partition\_id

LEFT OUTER JOIN syscolumns ON syscolumns.id = partitions.object\_id

 AND syscolumns.colid = cols.partition\_column\_id

WHERE allocunits.allocation\_unit\_id = 72057594039828480

ORDER BY nullbit

INSERT INTO @schema

SELECT -3, 1, 'StatusBitsA', 0, 0, 0, 0, 2147483647, 0, 0

INSERT INTO @schema

SELECT -2, 1, 'StatusBitsB', 0, 0, 0, 0, 2147483647, 0, 0

INSERT INTO @schema

SELECT -1, 2, 'LenFixedBytes', 52, 0, 10, 0, 2147483647, 0, 0

SELECT

s.\*,

CASE WHEN s.leaf\_offset > 1 AND s.bitpos = 0 THEN

SUBSTRING

(

@RowLogContents,

ISNULL((SELECT TOP 1 SUM(x.length) FROM @schema x WHERE x.[column] < s.[column] AND x.leaf\_offset > 1 AND x.bitpos = 0), 0) + 1,

s.length

)

ELSE

SUBSTRING

(

@RowLogContents,

(col.columnOffValue - col.columnLength) + 1,

col.columnLength

)

END AS hex\_string

FROM @schema s

LEFT OUTER JOIN @colOffsetTable col ON col.colNum = (s.leaf\_offset)

GO



마지막 결과셋을 보면 필드 속성정보 와 데이터도 hex정보로 출력 되는것을 확인 할 수 있습니다.
아래는 레코드구조입니다. 출력된 결과 와 아래 이미지 구조(결과)와 비교하면서 확인해보면
구조에 대한 분석이 훨씬 직관적일 것 같습니다.


참고: [http://www.sqlservercentral.com/articles/Transaction+Log/71415/](http://www.sqlservercentral.com/articles/Transaction%2BLog/71415/)