Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.27, NO.2, Apr. 2017 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2017.27.2.223

Microsoft SQL Server 삭제 이벤트의 데이터 잔존 비교

신 지 호†‡
경찰대학 국제사이버범죄연구센터

Comparison of Remaining Data According to Deletion Events on Microsoft SQL Server

Jiho Shin†‡
International Cybercrime Research Center, Korean National Police University

요 약

Microsoft SQL Server의 데이터 복구와 관련된 그간의 연구는 삭제된 레코드가 트랜잭션 로그에 존재하는 경우를 바탕으로 복원하는 방법이 주를 이루었다. 그러나 관련 트랜잭션 로그가 존재하지 않거나 물리 데이터베이스 파일이 Server와 연결되어 있지 않은 경우 적용하기 곤란한 한계가 있었다. 그러므로 범죄 과정에서 용의자가 delete 외 다른 이벤트를 이용하여 삭제할 가능성이 있기 때문에 이에 대한 삭제된 레코드의 잔존을 확인하여 복구 가능성을 확인해볼 필요가 있다. 이 논문에서는 Microsoft SQL Server 물리 데이터베이스 파일의 구조를 기초로 데이터 삭제가 수행되는 delete, truncate, drop 이벤트에 따른 Page 할당정보 유지 여부, 미할당 삭제데이터 존재 여부, 페이지 내 행오프셋 배열 정보의 변화를 실험하여 최종적으로 디지털포렌식 조사 시 관리도구 이용 가능성 및 데이터 복구 가능성을 확인하였다.

ABSTRACT

Previous research on data recovery in Microsoft SQL Server has focused on restoring data based on in the transaction log that might have deleted records exist . However, there was a limit that was not applicable if the related transaction log did not exist or the physical database file was not connected to Server. Since the suspect in the crime scene may delete the data records using a different deletion statements besides "delete", we need to check the remaining data and a recovery possibility of the deleted record. In this paper, we examined the changes "Page Allocation information" of the table, "Unallocation deleted data", "Row Offset Array" in the page according to "delete", "truncate" and "drop" events. Finally it confirmed the possibility of data recovery and availability of management tools in Microsoft SQL Server digital forensic investigation.

Keywords: Microsoft SQL Server, Database, Delete, Truncate, Drop, Recovery, Page

I. 서 론

디지털 사회로부터 넘쳐나는 데이터를 효과적으로 관리하고 편리하게 이용하기 위해 데이터베이스 시스템이 이용되고 있다. 범죄수사에서도 데이터베이스에 존재하는 다양한 형태의 자료를 분석하거나 삭제된 데이터를 복구하여 디지털 증거로서 사용하고 있다. 데이터베이스와 관련된 디지털포렌식 분야에서도 조사 대상에 혼재되어있는 데이터에서 수사에 필요한 정보를 효과적으로 찾아내는 "자료 분석(data analysis)"과 데이터베이스에서 삭제된 "자료를 복구(data recovery)"하는 두 가지 분야로 크게 나뉘

Received(01. 31. 2017), Modified(03. 14. 2017), Accepted(03. 14. 2017)
† 주저자, suchme@gmail.com
‡ 교신저자, suchme@gmail.com(Corresponding author)

어 있다[1]. 자료 분석 분야에서는 이미 빅데이터 기술, 데이터마이닝 등 여러 방법론 및 기술을 이용하여 자료(data)에서 정보(information)로의 가공을 가속하고 있다. 그러나 삭제된 데이터 복구 분야는 여전히 한계가 많다. 데이터베이스 시스템이 존재하는 플랫폼과 DBMS(Database Management System)의 종류에 따라 복구 탐지의 범위가 상이하고 성공 확률의 차이가 있기 때문이다.

데이터베이스의 데이터 삭제이벤트의 상당수는 흔히 알려진 delete문에 의해서 실행되며 이 수행 결과는 데이터베이스 트랜잭션 로그에 저장되므로, 데이터 복구와 관련된 연구 또한 이 트랜잭션(transaction) 로그를 이용한 연구에 치중되어 있다[2]. 그러나 이외에도 truncate 및 drop에 의해서도 데이터 삭제가 수행된다. delete 명령문은 주로 데이터베이스 사용자에 의해 데이터 가공 시 수행되며, truncate나 drop 명령문은 관리자에 의해 객체와 데이터를 초기화하거나 삭제하는 경우 수행된다. 이 같은 관점에서 데이터베이스의 각 삭제 이벤트에 따른 잔존 데이터의 차이를 비교해 볼 필요가 있다. 이와 같은 비교를 통해 도출된 복구 가능성은 디지털포렌식 조사 시 기술적 방향성 제시 측면에서 상당히 중요하기 때문이다. 이에 본 논문에서는 디지털포렌식 조사 시의 데이터 복구 관점에서 Microsoft SQL Server의 delete, truncate 및 drop 이벤트에 따른 레코드 잔존 여부를 실험하여 복구가능성을 확인하였다.

II. 조사실무와 선행연구

범죄 수사에서 압수·수색 대상이 데이터베이스인 경우 압수 대상 데이터를 현장에서 추출하거나 원본 압수의 형태로 진행된다. 현장에서의 자료 수색이 곤란하거나 수색한 결과 구하고자 하는 자료가 발견되지 않는 경우 절차에 따라 원본을 압수하는 경우가 있는데[1], Microsoft SQL Server인 경우 데이터베이스의 물리 원본 파일인 "〈데이터베이스 카탈로그 이름〉.mdf" 및 "〈데이터베이스 카탈로그 이름〉.ldf" 파일과 각종 로그 데이터를 복사하여 압수·제출 받아 조사관 또는 디지털포렌식 분석관의 분석용 컴퓨터에서 분석을 시도한다. 수집한 물리 원본파일과 트랜잭션 로그 등에 대한 상세분석을 통해서도 구하고자 하는 테이블 또는 자료가 발견되지 않는 경우에는 삭제된 데이터의 미할당 영역에 대한 조사가 진행된다.

Microsoft SQL Server의 데이터 복구와 관련된 선행연구와 기법으로는 로그파일에 남아있는 DML과 DDL의 동작에 대한 로그를 조사하는 트랜잭션에 의존한 복구 기법이 주를 이루었다. Theo Haerder와 Andreas Reuter는 삭제된 레코드가 트랜잭션 로그에 존재하는 경우를 바탕으로 삭제된 레코드를 복원하는 방법을 제시[2]하였으나, 트랜잭션 로그가 존재하지 않으면 불가능 하거나 조사 대상의 물리 데이터베이스 파일이 SQL Server와 연결되어 있지 않은 경우 적용하기 곤란한 한계가 있다. 결과적으로 삭제된 레코드 데이터가 데이터베이스 파일의 논리 페이지(Page)내에서 정리되지 않은 상태로 미할당 영역에 존재한다하더라도 이와 같은 논리적인 복구 방식으로는 한계가 존재한다. 이에 미할당 영역에 삭제된 데이터를 복구하는 여러 연구가 진행되었고, 이 기법에 따라 트랜잭션에 의존하지 않는 방식으로 데이터를 복구할 수 있게 되었다[3][4].

SQL Server 대상 디지털포렌식 조사 실무에서도 데이터베이스 물리 원본 파일에 대하여 사건 관련 키워드 검색을 거친 후, 발견된 내용이 존재한다면 그 오프셋을 기준으로 상세 분석을 시도하는 방식을 취한다. 이는 페이지에서 미할당 영역에 저장되어 있는 데이터를 검색하기 위함이다. 본 논문에서도 이와 같은 방식을 이용한 실험 비교를 진행할 것이다.

SQL Server에서 데이터 삭제 이벤트는 크게 조작어(DML: Data Manipulation Language)인 delete와 정의어(DDL: Data Definition Language)인 truncate, drop에 의해서 발생된다[5][6]. 이와 같은 이벤트 실행 후 데이터 잔존 여부를 비교하기 위해 테이블 객체 정보 변화, 실제 레코드가 저장되어있는 논리적 공간인 페이지 할당변화, 페이지 내부 존재하는 행오프셋 배열(row offset array)변화, 그리고 가장 중요한 실제 미할당 레코드 데이터의 잔존 유무를 실험을 통해 확인할 것이다. 이를 통해 디지털포렌식 조사 시 삭제된 레코드에 대한 복구 가능성을 확보하고, 데이터베이스 물리 원본파일에 대한 직접 탐지의 필요성 논하고자 한다.

III. Microsoft SQL Server

3.1 Microsoft SQL Server

Microsoft SQL Server는 Microsoft가 1989

사이베이스(Sybase)를 기반으로 개발한 관계형 데이터베이스이다[7]. 대량의 데이터를 효율적으로 관리하기 위해 여러 기업에서 Microsoft SQL Server를 이용하고 있다. Server 안정성과 GUI라는 편의성에 힘입어 초기 버전인 Microsoft SQL Server 1.0부터 최근 Microsoft SQL Server 2016까지 출시하며 계속적인 발전을 거듭했다[3].

Microsoft SQL Server에는 Catalog라 불리는 여러 데이터베이스가 서버에 연결되어 있고, 요구 서비스에 따라 각각의 Catalog 내 데이터가 제공된다. 데이터베이스 Catalog는 크게 시스템용과 사용자용으로 구분된다. 시스템 Catalog에는 데이터베이스를 운영하기 위한 여러 객체들로 구성되어 있으며, master, model, msdb, tempdb 등의 Catalog가 있다[3]. 사용자 Catalog는 이용하고자 하는 서비스의 목적과 필요에 따라 사용자가 정의·구성하여 생성한 데이터베이스를 말한다.

3.2 Microsoft SQL Server 데이터베이스 파일

Microsoft SQL Server 데이터베이스에서 관리하는 데이터 물리 원본 파일은 데이터파일과 로그파일로 구분된다. 데이터파일에는 실제 데이터가 저장되며 "〈데이터베이스 카탈로그 이름〉.mdf"으로 생성되고 최초 Catalog 생성 시 "C:\Program Files\Microsoft SQL Server\[SQL Server 버전]\MSSQL\DATA"(윈도우즈 OS 기준)에 저장된다. 로그파일은 데이터베이스의 객체정의, 데이터의 생성·수정·삭제 등과 같은 각종 이벤트와 트랜잭션에 대한 정보가 저장되며 Catalog 생성 시 "〈데이터베이스 카탈로그 이름〉.ldf"으로 데이터 파일과 함께 쌍으로 생성되는데, 기본 저장 경로는 데이터 파일의 위치와 동일하다.

데이터베이스 파일은 Fig. 1과 같이 "페이지"라는 논리적 구조체의 반복으로 구성되어 있으며, 수많은 페이지들이 B-tree 자료구조로 연결되어 데이터베이스 파일을 이룬다. 페이지는 Root 페이지, Index 페이지, Leaf 페이지로 구성된다. 각 인스턴스(Instance)의 데이터 요구(retrieving)에 의해 Root 페이지에 저장되어 있는 Index 페이지에 대한 정보를 읽어 호출하고, 호출된 Index 페이지는 실제 데이터가 저장되어 있는 Leaf 페이지를 호출하여 자료를 반환하는 구조를 가지고 있다.

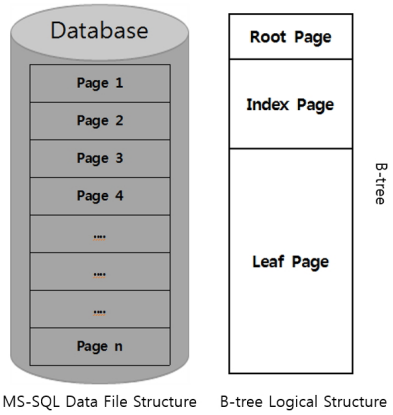

Fig. 1. Data file and logical structure of Microsoft SQL Server

통상 SQL Server 데이터베이스 조사에서 물리 원본 파일을 압수하여 분석하는 방식을 취하므로 SQL Server가 존재하였던 하드디스크 전체를 대상으로 조사하는 일은 드물다. 데이터베이스 파일 내 모든 논리 구조가 페이지를 이용한 B-tree 자료구조를 이용하여 연결되어있고, 데이터베이스 파일의 시작 또한 페이지로 구성되어 있기 때문에 데이터베이스 파일 자체의 고유 특성인 시그니쳐(Signature) 등을 고려할 필요는 없다.

3.3 Page Structure

Microsoft SQL Server의 논리적 구조를 이루는 페이지는 SQL Server에서 관리하는 기본 데이터 저장 단위이다[8]. 데이터 행을 쌓아놓은 페이지의 크기는 8,192byte로 1MB 당 총 128개의 페이지를 가질 수 있다. Fig. 2와 같이 페이지는 페이지 헤더(Page header), 데이터 행(Data row), 행오프셋 배열(Row offset array)로 구성되어 있다. 이와 같은 페이지 8개를 모아 익스텐트(Extent)라는 논리적 개념을 이용하여 페이지를 효율적으로 관리하는데 사용한다.

페이지는 데이터, 인덱스, 텍스트 및 이미지, 전역 할당 맵, 공유 전역 할당 맵, 페이지의 사용 가능한 공간, 인덱스 할당 맵, 대량 변경 맵, 차등 변경 맵 등의 종류로 유형화되어 있다. 그러나 본 논문에서는 삭제 이벤트 별 데이터의 잔존 여부를 확인할 것이므로 실제 데이터가 존재하는 데이터 페이지 외 다른 페이지 종류에 대한 논의는 생략하기로 한다.

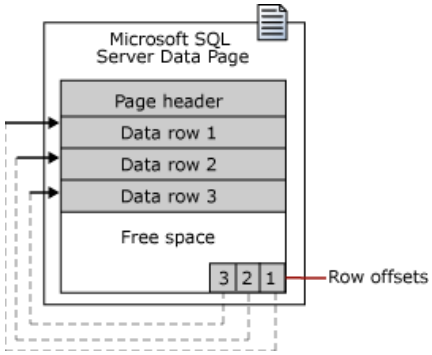

Fig. 2. Page structure (Reference: Microsoft Technet)

기본적으로 한 페이지의 단일 행의 데이터는 텍스트 및 이미지 데이터 타입을 제외하고 최대 8,060byte까지 저장될 수 있다. 만약 전체 행의 크기가 8,060byte를 초과하는 데이터 행인 경우 SQL Server는 행 내 가변길이 데이터의 열을 다른 페이지로 동적으로 옮기는 "대용량 행 지원" 작업을 수행한다[8].

페이지 헤더에는 해당 페이지의 번호와 종류 등의 정보로 구성되어 있다. 페이지 헤더 이후 실제 데이터 행이 순차적으로 저장되며, 모든 행이 끝난 이후에는 사용 가능한 공간인 빈공간(free space)이 위치하고 그 뒤로 2byte 크기로 구성된 행오프셋이 배열 형태로 존재한다.

3.3.1 Page Header

페이지 헤더는 각 페이지의 고유정보를 담고 있는 구조체로 매 페이지마다 존재하며 96byte의 크기로 구성되어 있다. 헤더 영역에는 Table 1와 같이 페이지의 여러 정보를 담고 있는데, 해당 페이지 고유번호, 페이지 종류, 페이지 내 존재하는 데이터 행의 개수, 자료구조 유지를 위한 이전 및 다음 페이지의 고유번호 등으로 구성되어 있다.

3.3.2 Data Row

데이터 행 영역은 실제 데이터 레코드가 저장되어 있는 구조체 영역으로 헤더가 끝난 이후부터 아랫방향으로 순차 저장된다. 각 행의 시작에는 행 헤더(Row header)가 위치하게 되며 이후 고정길이 데이터와 가변길이 데이터가 순차적으로 저장된다. 이때

Table 1. Structure of page header

Offset	Content
00	Header Version
01	Type
02	Type Flag Bits
03	Level
04-05	Flag Bits
06-07	Index ID
08-11	Previous Page ID
12-13	Previous File ID
14-15	Pminlen
16-19	Next Page ID
20-21	Next PageFile ID
22-23	Slot Count(Row Count)
24-27	Object ID
28-29	Free Count
30-31	Free Data
32-35	Page ID
36-37	File ID
38-39	Reserved Count
40-43	Lsn1
44-47	Lsn2
48-49	Lsn3
50-51	Xact Reserved
52-55	XdesID Part2
56-57	XdesID Part1
58-59	GhostRec Count
60-95	Reserved

데이터 행에서 가변길이 데이터를 관리하기 위해 가변길이 데이터의 오프셋을 별도로 관리한다. 각 행에서 고정길이 데이터를 먼저 저장하고 가변길이 데이터의 오프셋을 관리하는 이유는 저장 공간을 효율적으로 관리하기 위함이다. 데이터 행에 대해서는 페이지 구조를 전체적으로 알아본 후 자세히 알아보기로 한다.

3.3.3 Row Offset

페이지의 마지막 영역에 존재하는 행오프셋 배열은 페이지 내에서 데이터 행의 시작 오프셋 정보를 저장하고 있다. 각 행오프셋은 데이터 열 당 2byte의 크기로 저장되며 Fig. 2에서 보는 바와 같이 페

이지의 마지막부터 값을 채워 나간다. 데이터 페이지의 경우, 헤더가 끝난 지점부터 데이터 행이 시작되므로 항상 모든 행오프셋은 '0x60'으로 시작된다. 다시 말해, Microsoft SQL Server는 LEO(Little Endian Ordering)의 저장방식을 취하므로 모든 페이지의 마지막 값은 "0x6000"이 된다.

행오프셋을 이용하여 데이터 행의 크기를 계산할 수도 있다. 행오프셋은 순차적인 정보이므로, "1st 데이터 행의 크기 = 2^{nd} 행오프셋 - 1^{st} 행오프셋"과 같은 계산으로 데이터 행의 크기를 구할 수 있다.

3.4 Data Row Structure

데이터 행은 헤더, 고정길이 데이터(Fixed Length Data), 부가정보, 가변길이 데이터(Variable Length Data)로 구성된다(Table 2 참조). 고정길이 데이터가 먼저 위치하는 이유는 길이가 정해진 데이터를 먼저 저장하여 페이지 내 공간을 효율적으로 사용하기 위함이다. 더불어 Null 여부를 관리하기 위해 Null bitmap 정보를 별도로 두고 있으며, 가변길이 열의 데이터를 효율적으로 관리하기 위해 가변길이 데이터의 오프셋 정보를 이용한다.

Table 2. Row structure

Size(byte)	Content
1	Status Bits 1
1	Status Bits 2
2	Length of fixed length portion of in the record
n	Fixed length data
2	Total number of columns in the record
n	Null bitmap
2	Number of variable length column
Number of variable length column × 2	Variable column offset array
n	Variable length data

3.4.1 Data Row Header

데이터 행 헤더는 Status Bit1, Status Bit2를 표현하기 위해 각 1byte를 사용하여 Bit로 정보를 관리한다. 다음으로 위치하는 정보는 Row의 시작부터 고정길이 데이터까지의 길이정보를 2byte를 이용하여 표현한다. 즉, 행의 시작부터 고정길이 데이터가 끝나는 지점의 크기 정보를 기록하여 이 영역의 크기를 표시한다.

3.4.2 Fixed Length Data

고정길이 데이터 영역은 테이블 설계 시 결정된 데이터 타입(data type)에 따라 고정길이형 데이터가 저장되는 영역이다. 숫자형으로는 INT, 문자형으로는 CHAR가 대표적인 고정길이 데이터 타입이다. Fig. 3에서도 알 수 있듯이 고정길이 데이터가 해당 행에서 먼저 위치함을 확인할 수 있다. 고정길이 데이터 영역은 Null 값 또한 해당 영역을 차지하는 특징이 있다.

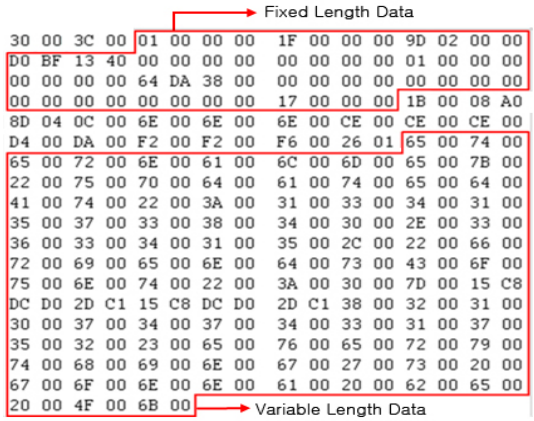

Fig. 3. The areas of fixed length data and variable length data in row

3.4.3 Variable Length Data

고정길이 데이터의 영역 이후 전체 열 개수정보, Null bitmap, 가변길이 열의 개수 정보 등 가변길이 데이터를 표현하기 위한 부가정보들이 위치한다. 이어 가변길이 데이터 오프셋 배열 정보가 위치하는데 오프셋은 각 가변길이 데이터의 마지막 위치를 표현한다. Fig. 3에서는 데이터 오프셋 배열의 마지막 값이 "0x126"인 것을 알 수 있는데, 가변길이 데이터의 오프셋 배열은 행 내 각 데이터의 마지막 위치를 표현하므로 전체 행의 크기가 294byte 라는 것을 유추할 수 있다. 또한 가변길이 데이터 영역의 특

징은 고정길이 데이터와는 다르게 Null 데이터를 저장하지 않는다는 특징이 있다.

IV. 데이터 삭제 이벤트

앞서 살펴본 바와 같이 T-SQL에서 데이터를 삭제하는 주된 방식은 데이터 조작어를 이용한 방법, 데이터 정의어를 이용한 방법으로 크게 나눌 수 있다. DML에 의한 데이터 삭제는 delete문을 이용하며 DDL에는 truncate와 drop이 있다.

4.1 Delete에 의한 삭제

delete는 조건절 where를 이용하여 조건에 부합하는 행들의 집합만을 삭제하거나 조건 없이 전체 행을 삭제할 수도 있는 데이터 삭제의 표준 구문이다[9]. 테이블 내의 행을 삭제하는 명령이므로 해당 테이블과 관련된 인덱스, 제약조건 등은 변화 없이 유지된다.

4.2 Truncate에 의한 삭제

truncate는 주로 테이블을 초기화 하는데 사용된다. 속도가 빠르며 테이블에서 모든 행을 삭제하는 효율적인 방법으로 알려져 있다[10]. 만약 자동으로 증가하는 열(seed column)이 해당 테이블에 존재한다면 이 값까지 초기화 된다. 이미 생성된 인덱스와 제약조건은 유지되지만 초기 테이블 설계 후 생성된 사용자에 의한 데이터는 모두 초기화 된다.

delete와의 가장 큰 차이점은, delete의 경우 먼저 해당되는 범위를 선택한 후 인덱스 페이지와 실제 데이터를 찾아내어 삭제하는 반면, truncate은 페이지를 관리하는 별도의 할당 페이지에 페이지가 다음 할당이 가능하다는 미할당 영역 표시를 하여 삭제를 하는 차이가 있다[11].

또 다른 중요한 특징은, truncate 명령이 로그를 사용하지 않는다는 것이다. 즉, 로그를 남기지 않는 명령문이므로 이미 알려진 트랜잭션 기반의 복원 기법으로는 데이터 복구가 불가능하다.

4.3 Drop에 의한 삭제

drop은 truncate과 같은 DLL의 한 종류로 주로 객체를 삭제할 때 사용된다. delete와 truncate와 같이 삭제 후 테이블 정보가 잔존하는 것과는 다르게 객체 자체를 삭제하기 때문에 테이블정보, 관련 인덱스 및 제약조건 정보를 모두 삭제하게 된다.

Table 3. Query commands of record deletion event

Command	Query
delete	DELETE FROM [object name] WHERE [condition]
truncate	TRUNCATE TABLE [object name]
drop	DROP TABLE [object name]

V. 삭제 데이터의 탐지 및 복구

5.1 데이터 탐색 및 키워드 조사

데이터베이스 원본을 대상으로 하는 디지털포렌식 조사 실무에서 데이터를 탐색하기 위해 Microsoft의 Management Studio, Oracle의 Toad 등 해당 데이터베이스에 접근 가능한 각종 관리 및 분석 툴을 이용하여 논리적인 데이터 조사를 먼저 수행한다. 그러나 논리적 데이터 영역에서 찾고자 하는 데이터가 존재하지 않는다면 삭제된 레코드 등 미할당 영역에 대한 조사를 수행해야 한다. 왜냐하면 데이터베이스 도구 등을 이용하여 추출한 논리적 데이터 결과 셋(set)과 각 페이지에 실재하고 있는 데이터에 차이가 있을 수 있기 때문이다. 삭제된 데이터의 경우 페이지 내 미할당 영역에는 존재하지만 관리도구 및 쿼리 실행 등의 논리 결과 셋에서는 제외될 수 있으므로 포렌식 조사 시 키워드 검색을 이용하여 미할당 영역을 포함한 실제 영역을 탐지할 필요가 있다.

검색에 사용될 키워드는 사건과 관련된 가장 연관성이 높은 키워드(unique value)를 선정하여 중복 데이터 검출을 피하는 것이 좋다. 키워드 검색에서 찾고자 하는 값이 검출되었다면 발견된 오프셋 주위의 데이터를 조사하여 데이터 존재 유무를 탐색해야 한다.

5.2 행오프셋 배열 탐지

키워드 검색 결과 해당 데이터가 페이지 내 존재한다면 그 페이지의 상세 탐지가 필요하다. Microsoft SQL Server 페이지는 앞서 살펴본 바와

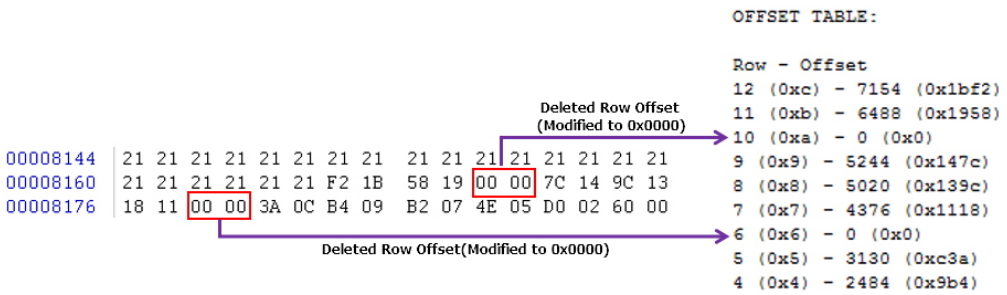

Fig. 4. Transformed row offset array according to row deletion

같이 "페이지 헤더-데이터 행-빈공간-행오프셋 배열"로 구성되어 있으므로 행오프셋 배열 영역을 탐지하여 탐지된 데이터가 삭제되었는지 여부를 조사할 수 있다.

먼저 Microsoft SQL Server 관리도구를 이용한 접근이 가능한 경우라면 해당 키워드가 발견된 위치의 페이지의 ID를 확인한 후 Transaction-SQL의 DBCC(Database Console Command) 명령어를 이용해 해당 페이지의 헤더와 실제 데이터, 그리고 행오프셋 배열을 확인할 수 있다. 만약 조사대상 원본이 훼손되어 관리도구를 통한 정상적인 접근이 어렵다면 Hex Editor를 이용한 직접 조사도 가능하다.

데이터의 헤더에서 명시하고 있는 행의 갯수가 몇 개인지, 만약 발견된 위치의 데이터가 삭제이벤트에 의해서 삭제된 데이터라면 Fig. 4와 같이 행오프셋 값을 "0x0000" 으로 갱신하므로 이를 확인하여 삭제된 데이터의 유무를 확인해야 한다. 다만, Microsoft SQL Server의 버전에 따라 행오프셋을 이전 행의 오프셋 값을 덮어씌우는 방식으로 갱신하는 경우도 있으므로 유의할 필요가 있다.

5.3 삭제 Row 데이터 복구

순차적으로 저장된 데이터 행 중 1개의 행이 삭제된 경우 Fig. 4와 같이 이전 행과 다음 행의 행오프셋 값을 이용하여 삭제된 행의 크기를 구할 수 있다. 만약 삭제된 행이 연속되어 존재한다면 첫 번째 삭제된 행 데이터의 구조를 조사한 뒤 순차적으로 행 데이터를 복구해 나가야 한다.

삭제된 Row의 전체 영역을 확인한 후에는 각 컬럼 별 데이터를 분리하여 복구해나가야 한다. 이를 위해서 Table 2의 Row Structure를 참조하여 각 컬럼 별 파싱과정을 거쳐 데이터를 구할 수 있다.

고정길이 데이터타입의 경우 정해진 길이만큼 데이터를 분리하여 카빙(carving)하면 가능하다. 가변길이 데이터의 경우에는 각 가변길이 컬럼 데이터에 대한 Null bitmap 및 오프셋 정보들을 먼저 파악한 후 이에 따라 가변길이 컬럼의 데이터를 분리하여 카빙하면 전체 행 데이터를 복구할 수 있다.

VI. 실험 결과

6.1 도구 선정과 실험 방식

실험에서는 5장에 기술된 삭제된 레코드 탐지 및 복구 기법을 토대로 각 삭제 이벤트에 따른 레코드 잔존 여부 내용을 확인하였다.

현재 Microsoft의 지원정책 상 Microsoft SQL Server 2008 이전에 출시한 하위 버전은 개발사의 지원이 종료된 상태이다[7]. 또한 앞서 살펴본 데이터의 저장, 관리 등에 필요한 각 요소와 구조는 버전 2008부터 2016까지 모두 동일한 형태를 가지므로, 개발사 기술 지원이 가능하여 향후에도 사용될 가능성이 있는 제품 중 출시일에 따른 가장 폭넓은 사용층을 가진 2008 버전을 실험 대상으로 선정하였다.

실험 방식은 다음과 같다. 먼저 Microsoft SQL Server에 새로운 카탈로그를 생성한 후 1개의 Integer 타입 컬럼과 3개의 Varchar 타입 컬럼으로 구성된 테이블을 생성하였다. 테이블을 생성한 후 임의의 20개 데이터 행을 삽입하였다. 이후 Detach 작업을 통해 원본 mdf, ldf 파일을 백업해 보관하고 실험 시마다 백업해놓은 원본 mdf, ldf 파일을 SQL Server에 Attach하여 이용하였다. 기존의 연구에서는 주로 delete 명령문 수행 이후 삭제 이벤트에 대한 트랜잭션 로그를 대상으로 실험을 진행한 반면, 본 실험에서는 Table 3의 각 삭제 쿼리를 수행하였고 이후 페이지 내 데이터의 존재 유무 등을

	PageFID	PagePID	IAMFID	IAMPID	iam_chain_type	PageType
1	1	79	NULL	NULL	In-row data	10
2	1	78	1	79	In-row data	1

Fig. 5. Page allocation information of the experiment table(before deletion)

```
4747DFD0:   00000000 00000000 ee02cc02 aa028702
4747DFE0:   63024202 2202ff01 dd01ba01 90016d01
4747DFF0:   48012501 0501e500 c600a600 84006000
```

Fig. 6. Row offset array of PagePID 78 (before deletion)

```
4747C060:   30000800 04000000 04000003 001a0020
4747C070:   0024004d 696b6165 6c61466f 776c6572
4747C080:   31393637 30000800 05000000 04000003
4747C090:   0019001e 00220043 6f72796e 6e547365
4747C0A0:   6e673139 35393000 08000600 00000400
4747C0B0:   00030019 001c0020 00416c79 7368614b
4747C0C0:   696d3139 36393000 08000700 00000400
4747C0D0:   00030018 001b001f 00417665 72794b69
4747C0E0:   6d313939 30300008 00090000 00040000
```

Fig. 7. Starting Area of Row data in the Page(PagePID 78) (before deletion)

살펴보는 방식으로 진행하였다.

실험에 앞서, 데이터 삽입 후 페이지 할당 정보, 행오프셋 배열, 행 데이터의 상태를 확인하였다. 실험 테이블에는 2개의 페이지가 할당되어 있고 실제 행 데이터는 PagePID 78 페이지에만 저장되어 있으며(Fig. 5 참조) 헤더와 행오프셋 배열 상 데이터는 정상적으로 저장되어 있음을 확인하였다(Fig. 6, Fig. 7 참조).

삭제 이벤트 수행 후 살펴본 변화 내역은 다음과 같다. 먼저 이벤트 수행 후 객체의 존재유무를 살펴보았다. 그리고 페이지 할당 정보의 변화 유무를 확인하고, 미할당 영역 내 데이터 잔존 유무와 페이지 내 행오프셋 배열의 초기화 여부를 살펴보는 순으로 진행하였다. 이를 바탕으로 데이터베이스 조사 시 관리도구 이용가능성을 확인하고 최종적으로 데이터 복구 가능성을 확인하였다.

6.2 이벤트별 실험 결과

6.2.1 Delete

실험에 사용한 명령문은 "DELETE FROM Employee"로 별도의 조건을 붙이지 않고 전체 행을 삭제하였다. 실험 결과 삭제 후 테이블 객체와 페이지 할당은 변화 없이 동일하게 유지되었다. 행 데이터가 저장되어있는 PID 78 페이지를 탐지한 결과 행오프셋 배열은 초기화 되었으나 미할당 삭제 데이터가 페이지 내 그대로 존재하고 있음을 확인하였다.

6.2.2 Truncate

실험에 사용한 명령문은 "TRUNCATE TABLE Employee"로 테이블 전체를 초기화 하였다. 실험 결과 truncate를 이용한 테이블 초기화 후 테이블 객체는 계속적으로 유지되었으나 페이지 할당 정보 자체가 초기화 되어 조회되지 않았다. 행 데이터 존재 유무를 확인하기 위해 PID 78 페이지의 Raw 데이터를 직접 탐지한 결과, 행오프셋 배열 및 행 데이터 모두 변화 없이 그대로 존재하고 있음을 확인하였다, truncate 수행 시 페이지 할당정보 관리 페이지에서 해당 테이블 객체와 페이지의 링크를 해제하여 미할당으로 표시하는 방식임이 확인 되었다. 다만 이 경우 관리도구 통해 테이블 객체에 할당된 페이지 정보를 확인하기 어려우므로 키워드 검색 등을 통한 직접 조사를 수행해야 한다.

6.2.3 Drop

실험에 사용한 명령문은 "DROP TABLE Employee"로 테이블을 삭제하였다. 실험 결과 drop은 테이블 객체를 삭제하는 명령어이므로 객체는 삭제와 동시에 페이지 할당 정보도 초기화 되었다. 객체 자체가 삭제되어 관리도구를 통한 페이지

Table 4. Experiment result of data remaining according to deletion events

	Table Object	Page Allocation Info.	Unallocation Deleted Data	Row offset Info. in Page	Management Tool Usability	Data Recovery Possibility
DELETE	maintain	maintain	exist	initialization	possible	possible
TRUNCATE	maintain	initialization	exist	maintain	impossible	possible
DROP	removed	initialization	exist	maintain	impossible	possible

탐색이 어려우므로 PID 78 페이지의 Raw 데이터를 직접 탐지하였다. 그 결과 truncate과 동일하게 행오프셋 배열 및 행 데이터 모두 변화 없이 그대로 존재하고 있음을 확인하였다.

6.3 정리

실험 결과(Table 4 참조)를 정리하면 다음과 같다. delete는 DML의 하나로 명령어 수행 후에도 객체 자체와 페이지 할당정보, 미할당 삭제 데이터는 그대로 유지되었다. 그러나 행오프셋 정보는 삭제되어 "0x0000"으로 갱신됨을 확인하였다. 결과적으로 delete를 삭제 수행 시 페이지 할당정보는 변화 없이 유지되므로 조사 시 관리도구 이용이 가능하며, 미할당 삭제데이터를 대상으로 데이터 복구가 가능하다. truncate는 DDL의 하나로 명령어 수행 후에 테이블 객체와 미할당 삭제 데이터, 행오프셋 정보는 그대로 유지되었다. 그러나 페이지 할당정보가 초기화되어 조사 시 관리도구 이용이 불가능한 것으로 확인되었다. 다만 미할당 삭제 데이터가 유지되고 있으므로 키워드 조사와 같은 직접 탐지의 방법으로 데이터 복구가 가능하다. drop은 truncate과 다르게 객체 자체를 삭제하는 방식을 취하므로 명령어 수행 후에 테이블 객체 자체와 페이지 할당 정보는 삭제되어 조사 시 관리 도구 이용이 불가능한 것으로 확인되었다. 그러나 테이블 객체와 페이지 할당 정보만 삭제할 뿐 페이지 자체를 초기화 하지 않으므로 미할당 삭제 데이터와 페이지 내 행오프셋 정보는 그대로 존재하므로 truncate과 같이 직접 탐지의 방법으로 데이터 복구가 가능한 것으로 확인하였다.

VII. 결 론

SQL Server 데이터베이스 삭제된 레코드와 관련된 그간의 연구는 트랜잭션 로그를 이용한 연구가 주를 이루었다. 그러나 디지털포렌식 조사 시에는 다양한 방법에 의한 데이터의 고의 삭제 가능성을 고려해야 하므로 트랜잭션 외 물리 데이터베이스 파일의 미할당 영역에 잔존하는 데이터의 잔존 여부를 반드시 확인해야 한다.

본 논문에서는 Microsoft SQL Server 물리 데이터베이스 파일의 구조를 분석하여 데이터가 존재하는 영역을 살펴보고, 이를 기초로 데이터 삭제가 수행되는 주요 이벤트인 delete, truncate, drop 명령문에 따른 ① 테이블 객체의 변화, ② 페이지 할당 정보 유지 여부, ③ 미할당 삭제데이터 존재 여부, ④ 페이지 내 행오프셋 정보의 변화를 확인하였다. 이를 통해 Microsoft SQL Server 데이터베이스에 대한 디지털포렌식 조사 시 ⑤ 관리도구 이용 가능성과 최종적으로 ⑥ 데이터 복구 가능성을 실험하였다.

실험 결과 세 종류의 삭제 이벤트로 인한 Page 할당 정보 유지 및 행오프셋정보 등의 변화는 다소 차이가 있는 것으로 확인되었으나, 결과적으로 미할당 삭제 데이터는 모두 복구 가능한 것으로 확인되었다. 이는 디지털포렌식 조사를 수행함에 있어 Microsoft SQL Server 물리 데이터베이스 파일에서 의미있는 정보를 추출하는데 도움이 될 것이다. 다만 이러한 결과를 실무에서 적용 가능하도록 도구 개발을 통해 효율적인 디지털포렌식 조사가 이루어질 수 있도록 해야 할 것이며, update 명령 또는 컬럼 삭제 행위와 같은 데이터 갱신 및 유실 이벤트와 관련된 연구는 향후 과제로 남을 것이다.

References

[1] Korean Police Investigation Academy, "The Advanced Course of Digital Forensic", KPIA, Jun. 2013.

[2] Theo Haerder and Andreas Reuter, "Theo Principles of Transaction-Oriented Database Recovery," ACM Computing Surveys (CSUR), vol. 15, no. 4, pp. 287-317, 1983.

[3] Park Soo-Young, "A Research for Record Recovery Method in Database," the Degree of Master Thesis, Korea University, Dec, 2013.

[4] Ryu Gi-Hwan, "A Study for Recovering Records of Microsoft SQL Server's Database," Degree of Master Thesis, Korea University, Dec, 2014.

[5] Microsoft Technet, Data Manipulation Language (DML) Statements (Transact-SQL), https://technet.microsoft.com/en-us/library/ff848799(v=sql.110).aspx

[6] Microsoft Technet, Data Definition Language (DDL) Statements (Transact-SQL),

https://technet.microsoft.com/en-us/library/ff848799(v=sql.110).aspx
〔7〕 WIKIPEDIA, Microsoft SQL Server, https://en.wikipedia.org/wiki/Microsoft_SQL_Server
〔8〕 Microsoft Technet, Understanding Pages and Extents, https://technet.microsoft.com/ko-kr/library/ms190969(v=sql.105).aspx
〔9〕 Itzik Ben-Gan, Microsoft SQL Server 2012 T-SQL Fundamentals, 1st ED., Microsoft Press, Jul. 2012.
〔10〕 Microsoft Technet, Deleting All Rows by Using TRUNCATE TABLE, https://technet.microsoft.com/ko-kr/library/ms188249(v=sql.105).aspx
〔11〕 Son Ho-Sung, Deep inside T-SQL Query Technique, 1st ED., Youngjin, Jul. 2004.

〈저자소개〉

신 지 호 (Jiho Shin) 정회원
2015년 2월: 고려대학교 정보보호대학원 디지털포렌식학과 석사
2008년 7월~2011년 7월: 경기시흥경찰서 사이버범죄수사팀 수사관
2011년 7월~2015년 1월: 경기남부지방경찰청 디지털증거분석실 분석관
2015년 1월~현재: 경찰대학 국제사이버범죄연구센터 선임연구원
〈관심분야〉 사이버범죄, 디지털포렌식