*Journal of The Korea Institute of Information Security & Cryptology* ISSN 1598-3986(Print)
VOL.28, NO.4, Aug. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.4.859

# Microsoft SQL Server의 원본 수집 방식에 따른 삭제 데이터의 복구 가능성 비교

신 지 호[†‡]
경찰대학 치안정책연구소

# Comparing Recoverability of Deleted Data According to Original Source Collection Methods on Microsoft SQL Server

Jiho Shin[†‡]
Police Science Institute, Korean National Police University

요 약

데이터베이스의 삭제된 데이터 복구와 관련된 그간의 연구는 삭제된 레코드가 트랜잭션 로그에 존재하는 경우이거나 데이터베이스 원본을 물리적 방식으로 수집하여 삭제 데이터의 탐지와 복구를 진행하는 방식이 주를 이루었다. 그러나 트랜잭션 로그가 존재하지 않거나 증거수집 현장에서 물리적 방식의 원본수집이 불가능한 경우 즉, 데이터베이스 서버 정지로 인해 피압수인이 업무상 손실 또는 권리침해를 이유로 거부하는 경우에는 적용하기 곤란한 한계가 있었다. 그러므로 증거수집 현장의 제약을 대비하기 위해 다양한 수집 방식을 살펴보고 이에 따른 삭제된 레코드의 복구 가능성에 대해서 확인해볼 필요가 있다. 이 논문에서는 Microsoft SQL Server 데이터베이스를 대상으로 하는 디지털포렌식 조사 시 원본 수집의 논리적·물리적 방식별로 수집된 원본 파일 내 삭제된 데이터의 잔존 여부를 실험하여 각 수집 방식에 따른 데이터 복구 가능성을 확인하였다.

ABSTRACT

Previous research related to recovering deleted data in database has been mainly based on transaction logs or detecting and recovering data using original source files by physical collection method. However there was a limit to apply if the transaction log does not exist in the server or it is not possible to collect the original source file because a database server owner does not permit stopping the database server because of their business loss or infringement at the scene. Therefore it is necessary to examine various collection methods and check the recoverability of the deleted data in order to handling the constraints of evidence collection situation. In this paper we have checked an experiment that the recoverability of deleted data in the original database source according to logical and physical collection methods on digital forensic investigation of Microsoft SQL Server database.

**Keywords:** Database, Microsoft SQL Server, Logical / Physical Collection, Deleted Data, Recovery, Page

## I. 서 론

현대사회의 방대한 데이터를 안전하고 효과적으로 저장·관리하고 또 이를 분석·활용하여 새로운 가치를 창출하기 위해 데이터베이스를 이용하고 있다. 디지털포렌식 조사에서도 데이터베이스에 적재된 데이터를 범죄수사에 활용하기 위해 데이터를 분석하거나 삭제된 레코드 데이터를 탐지하여 증거로 사용하고

Received(04. 16. 2018), Modified(1st: 05. 18. 2018, 2nd: 06. 27 .2018), Accepted(07. 16. 2018)

† 주저자, suchme@police.go.kr
‡ 교신저자, suchme@police.go.kr(Corresponding author)

있다. 이와 같이 데이터베이스와 관련된 디지털포렌식 분야는 크게 자료분석과 자료복구로 나뉘는데, 삭제된 데이터의 복구 분야는 데이터베이스 플랫폼과 DBMS 종류의 다양화와 클라우드 컴퓨팅 등 새로운 서비스플랫폼과 융합되어가는 양상으로 조사에 여전히 한계가 많다[1].

데이터베이스의 삭제된 데이터 복구와 관련된 그간의 연구는 주로 트랜잭션 로그를 대상으로 하거나[2] 물리적 방식으로 데이터베이스 원본을 수집한 후 이를 대상으로 직접 탐지하는 연구에 치중되어 있다[1]. 그러나 트랜잭션 로그를 이용하는 복구 방식은 트랜잭션 로그가 존재하지 않는 경우에는 곤란하다. 그나마 물리 원본 파일을 이용하는 복구 방식은 선행연구를 통해 삭제 데이터 잔존 유무를 확인할 수 있는 방법이 존재하지만, 원본 수집을 위해 서버를 정지시켜야 한다는 증거 수집 상 한계가 존재한다. 피압수인이 업무상손실 또는 권리침해 등을 이유로 서버 정지를 거부하는 경우에는 진행이 어렵기 때문이다. 이와 같은 관점에서 실행중인 서버를 정지하지 않고 수집된 원본을 대상으로 이루어진 삭제데이터 복구결과와 서버 정지 후 수집된 원본을 대상으로 이루어진 삭제데이터 복구에 대한 선행 연구결과를 비교하여 확인해볼 필요가 있다. 즉, 원본 수집에 있어 논리적·물리적 방식에 따른 삭제된 데이터의 복구가능성을 비교해볼 필요가 있다. 이와 같은 비교를 통해 도출된 데이터 복구 가능성은 향후 디지털포렌식 조사 시 범죄현장 또는 증거수집 현장에서 원본수집에 대한 상황별 방법 제시 측면에서 매우 중요하기 때문이다. 이에 본 논문에서는 논리적 방식과 물리적 방식으로 수집된 각 원본을 대상으로 삭제된 레코드의 존재 여부를 실험하여 비교하고 복구 가능성을 확인하였다.

## II. 데이터베이스 수집실무와 선행연구

범죄 수사에서 압수·수색 대상이 데이터베이스인 경우 범죄 관련 데이터를 현장에서 추출하거나 원본 압수의 형태로 진행한다[1]. 범죄 현장이나 압수수색과정에서 쿼리 실행 등 자료검색으로 범죄사실과 관련되어 있는 자료를 수색하게 되는데, 해당 자료가 발견되지 않을 경우 삭제되었을 경우를 가정하게 된다. 이때 데이터가 임의적으로 삭제된 정황이 발견된다면 우선 DML과 DDL 동작이 기록되는 트랜잭션 로그를 조사해야 한다. 수집한 트랜잭션로그 등에서도 원하는 자료가 발견되지 않는다면, 데이터베이스의 미할당 영역에 대한 조사가 진행해야 하는데 이는 데이터베이스 원본 파일을 대상으로 이루어진다.

Microsoft SQL Server 데이터베이스의 데이터 복구와 관련된 선행연구는 주로 DML과 DDL 동작이 기록되는 트랜잭션 로그에 관한 연구가 주를 이루었다. Theo Haerder와 Andreas Reuter(1983)는 삭제된 데이터와 관련된 내용이 트랜잭션 로그파일에 남아있는 경우 이를 기반으로 삭제된 레코드의 복원 방법을 제시했으나[2], 트랜잭션 로그를 저장하지 않거나 이를 수집하지 못하는 경우에는 불가능한 한계가 있다. 이러한 한계를 극복하기 위해서는 데이터베이스 원본을 대상으로 레코드 복구를 진행해야 한다. 박수영(2013)과 류기환(2014)은 선행연구를 통해 삭제된 레코드 데이터가 데이터베이스 원본 파일의 미할당 영역에서 정리되지 않은 상태로 존재할 수 있으므로[3][4] 직접탐지의 방식을 통해서 데이터 복구의 방안을 제시하였다. 그러므로 데이터베이스의 디지털포렌식 조사 실무에서도 원본 파일에 대한 해당 범죄사실과 관련된 키워드 조사를 수행하고, 관련 내용이 발견된 오프셋(offset)을 중심으로 상세 분석을 수행하게 된다.

이와 같이 미할당 영역의 삭제된 데이터를 조사하기 위해서는 데이터베이스의 원본을 대상으로 디지털포렌식 조사가 이루어져야 하는데, 원본을 수집하는 방식은 크게 두 가지로 나뉜다. 데이터베이스 관리 시스템에서 제공하는 백업 기능을 활용하여 원본 파일(.bak)을 새롭게 생성하는 '논리적 방식'과, 데이터베이스의 서비스와 연결된 원본 파일(.mdf, .ldf) 자체를 수사관의 백업장치에 복사하여 수집하는 '물리적 방식'이 존재한다.

본 논문에서는 데이터베이스의 원본을 대상으로 미할당 영역 내 존재하는 삭제된 레코드 데이터를 조사함에 있어 위 두 가지 원본 수집 방식에 따른 레코드 복구 가능성을 확인하고자 하였다. 이를 확인하기 위해 Microsoft SQL Server의 각 방식별 수집 원본을 대상으로 동일한 실험을 수행하였다. 즉, 논리적 방식과 물리적 방식에 의해 수집된 각 원본을 대상으로 SQL 삭제 이벤트 실행 후 데이터 잔존 여부를 비교하기 위해 페이지(page) 할당 변화, 페이지 내 행오프셋 배열 변화, 행 데이터 잔존 유무를 확인하여 복구 가능성을 확인한 것이다. 이 결과를 통해 디지털포렌식 조사 시 범죄 현장 또는 압수수색 현장에서 상황별 원본 수집 방식의 방향성을 제시하

고자 한다.

## III. Microsoft SQL Server[1]

### 3.1 Microsoft SQL Server

Microsoft SQL Server는 Microsoft가 1989 사이베이스(Sybase)를 기반으로 개발한 관계형 데이터베이스이다[5]. 서버 안정성과 GUI 편의성으로 개인 및 기업에서 폭 넓게 이용되고 있으며 최근에는 Microsoft SQL Server 2017까지 출시하였다. 또한 Linux 와 같은 이기종 플랫폼에서도 작동이 가능하도록 범용성을 확장하였으며 SQL Server에 연동되는 데이터 분석용 BI 솔루션, 머신러닝 서버, Analytics Platform 시스템 등을 개발하여 제공하는 등 발전을 거듭하고 있다[6].

### 3.2 Page Structure

Microsoft SQL Server 데이터베이스 파일의 논리적 구조를 이루는 "페이지(page)"는 SQL Server에서 관리하는 기본 데이터 저장 단위로 8,192byte로 구성되어 있으며 데이터, 인덱스, 텍스트 및 이미지 등 몇 가지 종류로 유형화 되어있다[7]. Fig. 1과 같이 페이지는 헤더(page header), 데이터 행(data row), 행오프셋 배열(row offset array)로 구성되어 있다. 헤더 영역에는 해당 페이지의 고유번호, 오브젝트ID, 다음 페이지 번호, 이전 페이지 번호, 저장된 행의 개수 정보 등으로 구성되어 있다. 데이터 행 영역은 실제 데이터 레코드가 저장되어있는 구조체 영역으로 헤더가 끝난 이후부터 아랫방향으로 순차 저장된다. 각 행은 "행 헤더(row header)-고정길이 데이터-가변길이 데이터" 순으로 저장된다. 마지막으로 행오프셋 배열은 페이지 내에서 각 행의 시작 오프셋 정보를 저장하고 있으며 각 행 오프셋은 데이터 행 당 2byte의 크기로 저장되며 Fig. 1에서와 같이 페이지의 마지막부터 값을 채워나간다. Microsoft SQL Server는 LEO(Little Endian Ordering)의 저장방식을 취하므로 모든 데이터 페이지의 마지막 값은 헤더 이후의 오프셋 값인 "0x6000"이 된다.

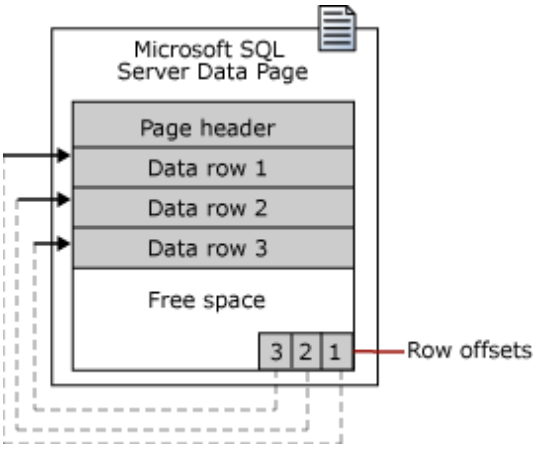


Fig. 1. Page structure (ref : Microsoft Docs)

### 3.3 Page Allocation Information

페이지는 논리적으로 Root, Index, Leaf로 구성되며 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 B-tree 자료구조를 가지고 있다[1]. 데이터베이스 내 객체가 생성되면 물리 데이터베이스 파일 내 특정 공간을 점유하여 객체와 연결된 페이지를 할당하게 된다.(Fig. 2 참조) 할당된 페이지는 데이터베이스 객체별로 관리되며 테이블 내 저장된 레코드 데이터 또한 페이지 내부에 구조적으로 저장되어 있다. 레코드의 호출 및 객체의 삭제와 관련된 SQL 이벤트 실행 시, SQL Server 엔진은 버퍼, 로그를 이용하여 이 페이지들을 대상으로 입·출력 및 초기화, 갱신을 수행한다.

| | PageFID | PagePID | NextPagePID | PrevPagePID | IAMFID | IAMPID | ObjectID | IndexID |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 115 | 0 | 0 | NULL | NULL | 213575799 | 0 |
| 2 | 1 | 114 | 0 | 0 | 1 | 115 | 213575799 | 0 |
| 3 | 1 | 832 | 833 | 0 | 1 | 115 | 213575799 | 0 |
| 4 | 1 | 833 | 834 | 832 | 1 | 115 | 213575799 | 0 |
| 5 | 1 | 834 | 835 | 833 | 1 | 115 | 213575799 | 0 |
| 6 | 1 | 835 | 836 | 834 | 1 | 115 | 213575799 | 0 |
| 7 | 1 | 836 | 837 | 835 | 1 | 115 | 213575799 | 0 |
| 8 | 1 | 837 | 838 | 836 | 1 | 115 | 213575799 | 0 |
| 9 | 1 | 838 | 839 | 837 | 1 | 115 | 213575799 | 0 |
| 10 | 1 | 839 | 840 | 838 | 1 | 115 | 213575799 | 0 |
| 11 | 1 | 840 | 841 | 839 | 1 | 115 | 213575799 | 0 |
| 12 | 1 | 841 | 842 | 840 | 1 | 115 | 213575799 | 0 |
| 13 | 1 | 842 | 843 | 841 | 1 | 115 | 213575799 | 0 |
| 14 | 1 | 843 | 0 | 842 | 1 | 115 | 213575799 | 0 |

Fig. 2. Page Allocation Information of Table on Microsoft SQL Server

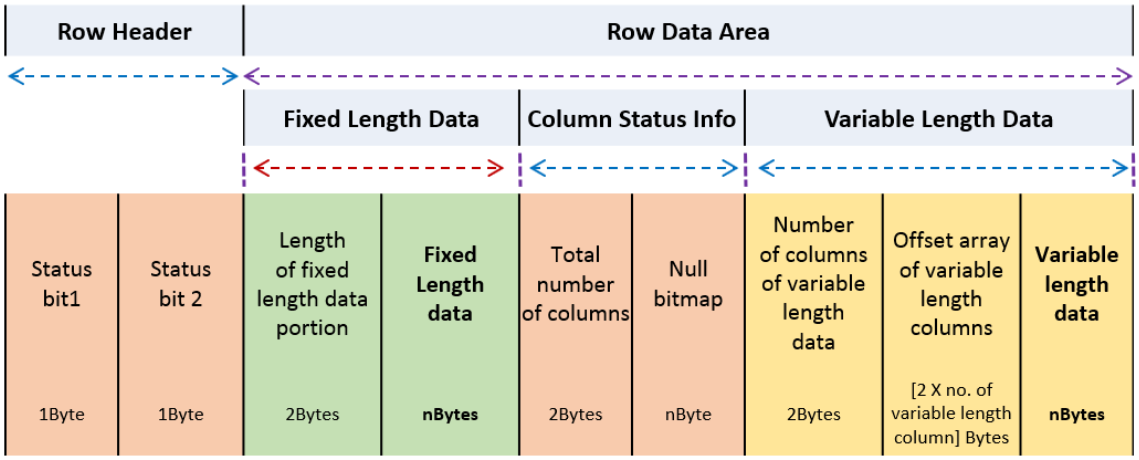


Fig. 3. Row structure of Microsoft SQL Server

### 3.4 Row Structure

페이지 내 순차적으로 저장되어 있는 Row 데이터는 크게 행 헤더, 행 데이터 영역으로 구성된다.(Fig. 3 참조) 행 헤더는 행 정보를 표현하기 위한 Status Bit 정보로, 레코드의 종류, 행 내 널비트맵 존재 유무, 가변길이 컬럼의 존재 유무 등의 정보를 비트로 표현한다. 이후 행 데이터 영역은 크게 고정길이 데이터 영역, 컬럼 상태정보 영역, 가변길이 데이터 영역으로 나뉜다. 고정길이 데이터(fixed length data) 영역은 전체 고정길이 데이터 영역의 크기 정보와 고정길이 데이터로 구성되며, 컬럼 상태정보 영역에서는 전체 컬럼의 개수 및 널비트맵을 표현한다. 가변길이 데이터(variable length data) 영역은 가변길이 컬럼의 개수와 가변길이 컬럼 데이터의 오프셋 배열 정보가 위치한 후 각 가변길이 데이터가 위치하게 된다. 가변길이 데이터가 나중에 위치하는 이유는 페이지 내 공간을 효율적으로 사용하기 위함이다[1].

### 3.5 삭제 Row 데이터의 탐지와 복구

T-SQL 명령문에 의해 삭제된 데이터를 탐지하는 방식은 크게 3단계로 나눌 수 있는데, 「검색 - 탐지 - 복구」의 순으로 수행된다. 먼저 활성 데이터 영역에서 select 등 쿼리 문장을 이용하여 데이터를 검색하고 구하고자 하는 데이터에 대한 조사를 수행하는 것이다. 구하고자 하는 데이터가 발견되지 않는다면 사건과 관련된 용의 키워드를 이용하여 키워드 조사를 해야 한다. 키워드 조사는 데이터베이스 원본 전체영역을 대상으로 수행되므로 빠짐없는 검색이 가능하다. 이후 키워드 조사에 의해 검색된 오프셋을 기준으로 페이지를 특정하고 도출된 페이지의 행 오프셋 배열을 조사하여 삭제된 데이터의 유무를 조사한다. 행 오프셋 배열은 해당 행이 삭제된 경우 0x0000 또는 이전  오프셋 정보로 갱신하므로 이를 기준으로 탐지가 가능하다. 삭제된 Row의 오프셋을 계산한 후 페이지 내 해당 위치로 이동하여 테이블 컬럼 scheme 정보와 비교하여 컬럼 별로 데이터를 카빙하면 복구가 완료된다.

### 3.6 Database Shrink

데이터베이스의 데이터 복구와 관계된 또 하나의 요소는 Microsoft SQL Server의 데이터베이스 축소(Database Shrink) 옵션이다. Shrink는 데이터베이스 물리 파일 끝에 있는 데이터 페이지를 파일 앞의 사용되지 않은 공간으로 이동하여 데이터 파일을 축소하여 공간을 재사용 가능한 영역으로 복구하는 작업으로, 파일 끝에 사용 가능한 공간을 충분히 확보한 다음 파일 끝에 있는 데이터 페이지의 할당을 해제하는 방식으로 작동된다[8]. 주기적으로 자동 실행되도록 하는 AUTO_SHRINK 속성 옵션이 존재하는데, 데이터베이스 설치 및 객체 생성 후 별도의 사용자 설정이 없는 경우 이 속성은 OFF 값으로 설정되어 있으므로 데이터 복구 관점에서 이에 대한 별도의 고려는 필요 없다. 다만 데이터베이스 관리자가 주기적으로 Shrink 명령을 시도하거나

AUTO_SHRINK 옵션이 TRUE로 설정된 경우에는 데이터베이스 객체에 할당된 페이지 중 물리 파일 끝에 존재하는 데이터 페이지의 할당을 해제하는 방식으로 초기화 될 수 있다.

## IV. 원본 수집 방식

Microsoft SQL Server 데이터베이스 원본 파일을 수집하는 방식은 크게 논리적 수집 방식(logical collection)과 물리적 수집 방식(physical collection)으로 나뉜다. 관점에 따라 크게 핫백업(hot backup)과 콜드백업(Cold bakup), 풀 백업(full backup)과 부분 백업(partial backup)으로 나누기도 하지만 이 논문에서는 논리적/물리적 백업 방식을 이용한 원본 수집 관점에서 삭제된 데이터의 복구 가능성을 비교하고자 한다.

### 4.1 논리적 수집 방식

논리적 수집 방식은 Microsoft SQL Server Engine에서 제공하는 "Back Up" 기능을 이용하는 방식을 말한다. 주로 SQL Server에 탑재되어 있는 관리도구를 이용하는 방식(Fig. 4 참조)으로, Enterprise Manager나 Management Studio에 백업 기능이 탑재되어 있어 기능을 활용할 수 있으며, T-SQL 쿼리를 직접 작성하는 방식(Table 1 참조)으로도 가능하다.

관리도구를 사용하는 경우 GUI를 통해 기능 진입이 가능하므로 매우 쉽고 간단하게 백업 수행이 가능하다는 장점이 있다. 수집하고자 하는 대상 카탈로그를 선택한 후 백업장치를 구성하면 해당 데이터베이스에 대한 백업이 완료된다.

논리적 방식은 Backup Type을 Full로 선택하는 경우 트랜잭션에 대한 백업이 함께 이루어지므로

Table 1. Query commands of database backup and restore

| Command | Query |
|---|---|
| backup | BACKUP DATABASE { *database_name* } TO 〈*backup_device*〉 [;] |
| restore | RESTORE DATABASE { *database_name* } FROM 〈*backup_device*〉[;] |

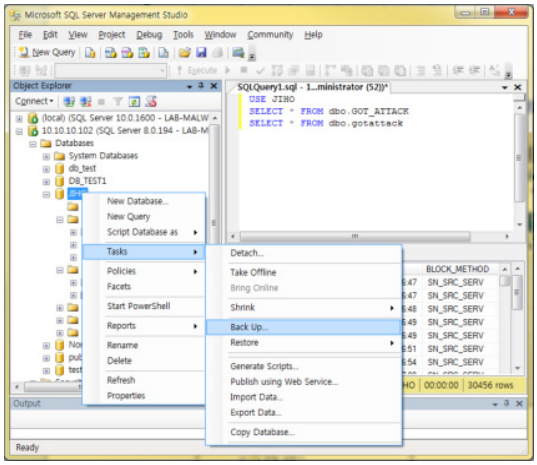


Fig. 4. Function of logical collection Method using Microsoft SQL Server Management Studio

트랜잭션 로그 백업에 대한 별도의 고려는 필요 없다. 데이터 분석을 위해 수집한 원본을 복원시키는 방법 또한 매우 간단한데, 복원할 대상 카탈로그를 선택한 후 "Restore" 기능으로 진입하여 수집된 원본을 선택하면 완료된다. 수집 대상 데이터베이스 서버에 관리도구가 설치되어 있지 않은 경우에는 GUI 방식이 아닌 T-SQL 쿼리를 이용하면 원본 수집과 복원이 가능하다 (Table 1 참조)

논리적 방식은 데이터베이스 서버 작동 중에도 해당 카달로그 전체 데이터를 수집할 수 있다. 서버 서비스를 정지하지 않아도 되기 때문에 범죄 수사 시 압수수색 현장에서 디지털포렌식 수사관의 부담을 덜 수 있다. 또한 .ldf에 저장되는 트랜잭션을 비롯한 기타 로그 정보도 데이터와 함께 한 번에 획득할 수 있다는 장점도 존재한다. 그러나 논리적 수집을 하기 위해서는 데이터베이스 관리자(sa) 또는 소유자(db_owner) 자격으로만 접근이 가능하므로 획득 대상 데이터베이스 관리자의 협조가 반드시 필요하다는 제약이 존재한다.

### 4.2 물리적 수집 방식

물리적 수집 방식은 데이터베이스와 연결된 물리 원본 파일을 직접 복사하는 방식으로 수집이 이루어진다. Microsoft SQL Server의 경우 데이터베이스의 물리 원본 파일인 "〈데이터베이스 카탈로그 이름〉.mdf" 및 "〈데이터베이스 카탈로그 이름〉.ldf" 파일을 제출받는 방식으로 이루어진다[1]. 물리 원본

파일을 수집하기 위해서는 우선 컴퓨터 디렉토리 내 해당 데이터베이스와 연결되어 있는 원본 파일의 위치를 특정하고 위 두 종류의 원본 파일을 수사관의 하드디스크에 복사하여 수집이 이루어진다. 통상 데이터베이스 물리 원본 파일은 별도의 사용자 설정이 없다면 Windows 운영체계 기준 "C:\Program Files\Microsoft SQL Server\MSSQL{버전번호}.MSSQLSERVER\MSSQL\DATA\"에 존재하므로 해당 디렉토리를 먼저 확인해 볼 필요가 있다.

물리적 방식으로 수집 시에는 SQL Server 서비스가 해당 물리 원본 파일을 점유하고 있으므로 서버를 정지하지 않거나 해당 카탈로그를 서버로부터 분리(detach)하지 않고 복사를 시도하면 점유위반(access violation)이 발생하게 된다. 그러므로 원본 파일을 복사하는 동안에는 반드시 데이터베이스 서버 서비스를 정지하거나 해당 카탈로그를 서버로부터 분리해야 한다는 제약이 존재한다. 만약 피압수인이 업무상 손실이나 권리침해 등으로 서비스 정지를 거부하는 경우에는 물리적 수집이 불가능하다는 한계가 있으므로 범죄현장이나 압수수색의 여러 상황에 맞는 데이터베이스 원본 수집 방식을 적절히 선택해야 할 필요가 있다.

앞서 살펴본 바와 같이 물리적 방식으로 수집 시 원본파일의 점유위반을 피하기 위해 서버를 정지해야 하는 제약이 존재하지만, 점유위반이 발생하지 않도록 데이터베이스의 dirty 상태에서도 원본 백업도 가능하다. 데이터베이스 서버는 예기치 않은 크래시(crash)에 대비하고 지속성을 담보하기 위해 로그(log), 데이터베이스 버퍼(database buffer), 데이터베이스 파일(database file) 등 3가지 요소를 이용하여 갱신에 대응하는 연속적인 동작이 수행된다[9]. 이때, 데이터베이스 버퍼에는 존재하지만 아직 데이터베이스 파일에 기록되지 않은 갱신 데이터 영역을 dirty page라 하며 이때의 데이터베이스를 dirty 상태라고 한다. 이를 활용하면 데이터베이스 서버를 정지하지 않은 상태에서도 원본 수집이 가능하다. 이와 같은 방식을 활용하여 원본을 수집하기 위해서는 마지막으로 커밋(commit)된 로그데이터와 최후 체크포인트까지의 원본, 그리고 수집한 dirty page를 조합하여 다시 읽을 수 있는 상태로 변환하는 복원 과정이 필요하다[10]. 다만, dirty 상태에서 원본을 수집할 경우 디지털증거의 수집에 있어 "원본동일성"의 흠결이 우려되고 이는 향후 공판과정에서 증거능력에 대한 논란의 소지가 있으므로 수집 원본을 피의사실에 대한 증거로 사용해야 한다면 피하는 것이 좋다. 더불어 dirty 상태에서의 백업은 앞서 살펴본 "동작중인 서버에서 원본을 수집"하는 논리적 수집 방식과 유사하므로, 물리적 방식이 곤란한 경우에는 Microsoft SQL Server에서 제공하는 공인된 기능인 "Back Up" 기능을 이용하여 clean 상태로 수집하는 것이 이와 같은 우려를 해소할 수 있다.

### 4.3 수집 방식 별 제약과 고려사항

앞서 알아본 원본 수집 방식별 제약사항을 정리해 보면 다음과 같다.

논리적 방식은 SQL Server가 제공하고 있는 기능을 활용할 수 있고 트랜잭션 로그의 수집이 동시에 이루어지므로 비교적 손쉽게 원본 수집과 복원이 가능하다. 그러나 데이터베이스 관리자(sa) 또는 사용자(db_owner)의 협조가 없이는 불가능하다는 제약이 존재한다. 이에 반해 물리적 방식은 데이터베이스 관리자의 협조 없이 전체 영역에 대한 원본 수집이 가능하지만 데이터베이스 서버를 일정시간 정지해야 하는 제약이 존재한다. 데이터베이스의 dirty 상태에서 원본을 수집하는 방식이 존재하지만, 논리적/물리적 수집방식을 모두 사용하지 못하는 경우에 한해 이용하는 것이 좋다. 앞서 살펴본 방식을 종합하여 범죄현장이나 압수수색 상황에 적절한 방식을 선택하는 것이 중요하다.

## V. 실험 결과

### 5.1 도구 선정과 실험 방식

앞서 살펴본 바와 같이 각 수집 방식별 제약 및 한계가 존재하므로 본 실험 결과는 범죄 현장이나 압수수색 상황에 적합한 수집 방법 설정에 활용될 수 있다. 실험에서는 4장에 기술된 각 수집 방식과 삭제된 데이터 탐지 방식을 기초로 T-SQL 삭제 이벤트 실행 이후 복구 가능한 레코드 잔존 여부를 확인하고 비교하였다. Microsoft SQL Server의 레코드 데이터 복구와 관련된 페이지 구조와 페이지 내 레코드 저장·관리 방식은 Microsoft SQL Server 2008버전부터 최근 출시한 2017버전까지 모두 동일한 형태를 가지므로[11] 현재 개발사의 기술 지원이

가능한 버전 중 가장 하위 버전인 2008 버전을 실험 도구로 선정하였다. 다만, 본 실험 결과의 적용가능성을 확보하기 위해 가장 최근 출시한 2017 버전에 대해서도 동일한 실험을 진행하여 결과를 확인하였다.

실험 대상의 데이터베이스가 설치된 서버컴퓨터의 운영체제 버전과 파일시스템, 데이터베이스의 실험대상 카달로그의 속성정보 (properties)를 정리하면 다음 Table 2와 같다.

실험 방식은 다음과 같다. 먼저 본 실험은 기존 물리적 수집 방식 후 데이터를 복구하였던 선행연구[1]의 확장된 실험으로, 선행연구에서 사용하였던 데이터를 활용하였다. 선행연구의 데이터는 Microsoft SQL Server의 데이터베이스 카달로그에 INTEGER 타입 컬럼 1개와 VARCHAR 타입 컬럼 3개로 구성된 테이블 객체이며, 이 테이블에 실험에 사용할 임의의 레코드 20개를 삽입하였다. 이후 DML 방식으로 데이터를 삭제하였다. 삭제 수행은 DELETE 명령어를 이용하여 조건 없이 전체 행을 대상으로 하였다. 이후 앞서 살펴본 각 방식으로 데이터베이스 원본을 백업하여 수집하였고, 각 원본을 다시 데이터베이스에 복원하여 페이지 할당 상태와 페이지 내 데이터의 상태를 확인하였다. 추가적으로, 삭제된 데이터의 복구와 밀접한 연관이 있는 데이터베이스 축소 기능에 대해서도 실험을 통해 함께 확인해 보았다.

삭제된 데이터의 복구와 기존의 연구에서는 물리적 방식만을 이용한 백업 실행 후 실험을 진행한 반면, 본 실험에서는 논리적·물리적 방식으로 백업을 진행하고 수집된 두 종류의 원본들을 이용하여 데이터 존재 유무를 살피는 방식으로 진행하였다. 선행연구의 결과와 본 실험의 결과를 종합적으로 고려하면 삭제 데이터의 복구 가능성 비교가 가능하다.

실험에 앞서, 데이터 삽입 후 정상 상태의 페이지 할당 정보, 행오프셋 배열, 행 데이터의 상태를 확인하였다. Fig. 5, Fig. 6, Fig. 7은 선행연구의 실험 결과로, 실험 테이블 객체에는 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)[1]. 삭제 이벤트 수행 후 물리적 방식으로 수집한 원본의 실험 결과는 기존의 연구 결과를 활용하였으며 논리적 방식으로 수집한 원본을 대상으로 ① 페이지 할당정보

Table 2. Properties of experiment server and database

| Category | Property | Value |
|---|---|---|
| Server Information | Operating System | Microsoft Windows NT 6.1 |
| | Platform | NT INTEL X86 |
| | File System | NTFS |
| | SQL Version | Microsoft SQL Server 2008 |
| Automatic | Auto Close | False |
| | Auto Create Statistics | True |
| | Auto Shrink | False |
| | Auto Update Statistics | |
| | Auto Update Statistics Asynchronously | False |
| Cursor | Close Cursor on Commit Enabled | False |
| | Default Cursor | GLOBAL |
| Miscellaneous | ANSI NULL Default | False |
| | ANSI NULLS Enabled | False |
| | ANSI Padding Enabled | False |
| | Arithmetic Abort Enabled | False |
| | Concatenate Null Yields Null | False |
| | Cross-database Ownership Chaining | False |
| | Date Correlation Optimization Enabled | False |
| | Numeric Round-Abort | False |
| | Parameterization | Simple |
| | Quoted Identifiers Enabled | False |
| | Recursive Triggers Enabled | False |
| | Trustworthy | False |
| | VarDecimal Storage Format Enabled | True |
| Recovery | Page Verify | CHECK-SUM |
| Service Broker | Broker Enabled | False |
| | Honor Broker Priority | False |
| | Service Broker Identifier | 02e73385-ed39-40b5-8666-1db52c7cbc99 |
| State | Database Read-Only | False |
| | Database State | NORMAL |
| | Encryption Enabled | False |
| | Restrict Access | MULTI USER |

| | 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 in 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 PagePID 78 (before deletion)

의 변화 유무, ② 페이지 내 행오프셋 배열의 변화 여부, ③ 미할당 영역 내 데이터 잔존유무를 살펴보았다.

## 5.2 수집 방식별 실험 결과

실험에 사용한 T-SQL 명령문은 "DELETE"로 별도 조건을 붙이지 않고 전체 행을 삭제하였다.

### 5.2.1 Logical Collection

논리적 방식에 의해 수집된 원본을 대상으로 실험한 결과, 행 삭제 후 페이지 할당은 변화 없이 동일하게 유지되었다. 페이지 내 행 오프셋 배열은 초기

| | 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. 8. Page allocation information of the experiment table according to logical collection method(after deletion)

```
476BDFD0:   00000000 00000000 00000000 00000000
476BDFE0:   00000000 00000000 00000000 00000000
476BDFF0:   00000000 00000000 00000000 00000000
```

Fig. 9. Row offset array in PagePID 78 according to logical collection method(after deletion)

```
476BC060:   30000800 04000000 04000003 0017001d
476BC070:   00210074 65737446 6f776c65 72313936
476BC080:   37393637 30000800 05000000 04000003
476BC090:   0019001e 00220043 6f72796e 6e547365
476BC0A0:   6e673139 35393000 08000600 00000400
476BC0B0:   00030019 001c0020 00416c79 7368614b
476BC0C0:   696d3139 36393000 08000700 00000400
476BC0D0:   00030018 001b001f 00417665 72794b69
476BC0E0:   6d313939 30300008 00090000 00040000
```

Fig. 10. Starting Area of Row data in PagePID 78 according to logical collection method(after deletion)

화 되었으나 페이지 내 미할당 영역에 행 데이터가 그대로 잔존하고 있음을 확인하였다.

### 5.2.2 Physical Collection

물리적 방식에 의해 수집된 원본을 대상으로 실험한 결과, 논리적 방식과 마찬가지로 행 삭제 후 페이지 할당에는 변화가 없었으며 행오프셋 배열은 초기화 되었고 행 데이터 또한 변화 없이 그대로 잔존하고 있음을 확인하였다[1].

### 5.2.3 Database Shrink

추가적으로 데이터베이스 축소 기능이 원본 수집에 영향을 끼치는지 알아보기 위해 논리적·물리적 수집 방식 모두에서 축소 기능 실행 후 실험을 수행하였다. 실험 결과 논리적·물리적 방식과 관계없이 데이터베이스 축소 기능을 수행하는 경우 페이지 할당 정보가 초기화 되었다. 미할당 영역 내 데이터는 2008 버전의 경우 논리적·물리적 방식 모두 데이터가 유지되었으며, 2017 버전의 경우에는 데이터베이스 축소 후 논리적 수집의 경우에만 미할당 데이터까지 초기화되는 것으로 확인되었다.

## 5.3 실험의 한계

본 실험은 특정 SQL Server 버전과 운영체제를 이용하여 제한적으로 수행되었으므로 다양한 운영체제의 종류, 데이터베이스 버전의 종류, 파일시스템의 종류 별 각 조합에 따른 데이터복구 성능비교 측면에서는 한계가 존재한다. 다만, 이 실험은 데이터베이스 각 버전 및 운영체제, 파일시스템, 각종 설정과 옵션 등의 조합별로 실험하여 복구속도, 복구율 등 성능평가를 목적으로 한 것이 아닌, 원본 수집 현장

Table 3. Experiment result of data recovery possibilities according to acquisition methods(Applied to ver. 2008 and 2017)

| Ver. | Collection Method | Database Shrink | Page Allocation Info. | Row Offset Info. in Page | Un-allocation Deleted Data | Recovery Possibility |
|---|---|---|---|---|---|---|
| 2008 | Logical | Non-shrink | maintain | initialized | exist | possible |
| | | Shrink | initialized | initialized | exist | possible |
| | Physical | Non-shrink | maintain | initialized | exist | possible |
| | | Shrink | initialized | initialized | exist | possible |
| 2017 | Logical | Non-shrink | maintain | initialized | exist | possible |
| | | Shrink | initialized | initialized | initialized | impossible |
| | Physical | Non-shrink | maintain | initialized | exist | possible |
| | | Shrink | initialized | initialized | exist | possible |

에서 디지털포렌식 조사관이 원본의 수집 방식을 선택하기 위한 방법 설정에 그 목적이 있다. 그러나 향후 데이터베이스를 대상으로 하는 디지털포렌식 분석에 활용되기 위해서는 향후 데이터복구 성능 평가에 관련된 연구가 후속될 필요가 있다.

### 5.4 정리

실험 결과를 정리하면 다음과 같다.(Table 3 참조) DELETE 명령 수행 후 논리적 수집 방식과 물리적 수집 모두 페이지 할당 정보는 유지되고 미할당 영역 내 행 데이터는 그대로 잔존하고 있었으나 페이지 내 행오프셋 배열은 초기화되었다. 두 방식 모두 원본 수집 전 데이터베이스 축소 기능을 수행하였더니 페이지 할당 정보는 초기화되었고, 미할당 영역 내 데이터의 잔존 여부는 논리적 방식의 원본 수집인 경우 데이터베이스 버전에 따라서 다른 결과를 보였다. 즉, 데이터베이스 축소 후 논리적 방식의 수집에 있어서 2008 버전의 경우에는 미할당 영역 내 데이터가 잔존하고 있으므로 직접 탐지와 복구를 통한 디지털포렌식 조사는 가능한 것으로 확인되었으나, 2017 버전의 경우에는 미할당 영역 내 데이터까지 초기화되는 것으로 확인되었다.

## VI. 결 론

Microsoft SQL Server 데이터베이스의 삭제된 레코드와 관련된 그간의 연구는 트랜잭션 로그를 이용하거나 물리 원본 파일 자체를 백업하는 방식을 이용한 연구가 주를 이루었다. 그러나 범죄 현장이나 압수수색의 상황에 따라 여러 제약이 발생할 수 있으므로 원본 수집에 대한 다양한 방법을 대비해 놓아야 한다.

본 논문에서는 Microsoft SQL Server의 두 가지 원본 수집 방식인 논리적 방식과 물리적 방식에 대해서 알아보았다. 데이터 복구와 관련된 요소로는 페이지의 구조, 원본 파일 내 페이지의 할당 방식, 데이터베이스 축소 기능을 살펴보았다. 이를 기초로 논리적·물리적 수집 방식에 따른 ① 페이지 할당 정보 유지 여부, ②페이지 내 행오프셋 정보, ③ 미할당 삭제데이터 존재 여부를 확인하였고, 추가적으로 ④ 데이터베이스 축소 수행으로 인한 삭제된 데이터 변화 여부도 실험을 통해 확인하였으며, 이 실험은 Microsoft의 기술지원이 가능한 최하위 버전인 2008과 최상위 버전인 2017을 대상으로 수행하여 데이터베이스를 대상으로 하는 디지털포렌식 현업에서의 적용가능성을 확보하고자 하였다.

실험 결과, 두 종류의 원본 수집 방식 모두 페이지 할당 정보와 미할당 영역 내 삭제된 데이터는 그대로 존재하였으며 행오프셋 정보는 초기화 되는 것을 실험으로 확인하였다. 더불어 데이터베이스 축소 기능 수행 시에는 논리적·물리적 원본 수집 방식 모두 페이지 할당 정보를 취소하였으며, 미할당 영역의 데이터 잔존 여부는 논리적 수집인 경우 데이터베이스 버전에 따라 다른 결과를 보였다. 이와 같은 결과는 디지털포렌식 조사를 수행함에 있어 원본 수집 시

현장 상황에 맞는 수집 방식을 제시하는데 도움이 될 것이다. 다만, Microsoft SQL Server의 데이터 복구에 관한 기존 연구와의 성능 비교나 공인된 데이터를 활용한 복구속도, 복구율과 같은 성능평가는 향후 과제로 남을 것이다.

## References


[1] Jiho Shin, "Comparison of Remaining Data According to Deletion Events on Microsoft SQL Server," Journal of The Korea Institute of Information Security & Cryptology, vol. 27, no. 2, pp. 223-232, 2017.

[2] Theo Haerder and Andreas Reuter, "Theo Principles of Transaction-Oriented Database Recovery," *ACM Computing Surveys (CSUR)*, vol. 15, pp. 287-317, 1983

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

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

[5] WIKIPEDIA, Microsoft SQL Server, https://en.wikipedia.org/wiki/Microsoft_SQL_Server, 2018.3.25

[6] Microsoft, Microsoft Data Platform, https://www.microsoft.com/en-us/sql-server/, 2018.3.25

[7] Microsoft TechNet, Understanding Pages and Extents, https://technet.microsoft.com/en-us/library/ms190969(v=sql.105).aspx, 2018.3.25

[8] Microsoft Docs, Shrink a Database, https://docs.microsoft.com/en-us/sql/relational-databases/databases/shrink-a-database, 2018.5.3

[9] Mark and Kimura Meiji, "Database first step: catch up with the database of the complete novice," Hanbit media, 2018.5.3

[10] Kim Kyung-Chang, "A Study on Data Backup and Recovery Technique for Dynamic Real-Time Database," Electronics and Telecommunications Research Institute, 1994

[11] Microsoft Docs, Pages and Extents Architecture Guide, https://docs.microsoft.com/en-us/sql/relational-databases/pages-and-extents-architecture-guide, 2018.6.30


## 〈저자소개〉


신 지 호 (Jiho Shin) 정회원

2015년 2월: 고려대학교 정보보호대학원 디지털포렌식학과 석사
2008년 7월~2011년 1월: 경기시흥경찰서 사이버범죄수사팀 수사관
2011년 7월~2015년 1월: 경기남부지방경찰청 디지털증거분석실 분석관
2015년 1월~2018년 1월: 경찰대학 국제사이버범죄연구센터 전임연구원
2018년 1월~현재: 치안정책연구소 과학기술연구부 연구관

〈관심분야〉 치안빅데이터, 사이버범죄, 디지털포렌식