데이터베이스의 삭제된 레코드 복구 기법의 한계

김동휘
고려대학교 컴퓨터정보통신 대학원
소프트웨어 보안학과
e-mail:k3126545@gmail.com

Limitations on the method of recovery for deleted record in Database

Dong-Hwi Kim
Dept of Software Security, Korea University

요 약

데이터베이스는 대용량의 데이터를 효율적으로 관리할 수 있는 장점을 가지고 있기 때문에 다양한 분야에서 사용되고 있다. 관련 범죄 발생 시 PC, 스마트폰 등 다지털 기기의 정보를 수집, 복구, 분석해 활용하는 '디지털포렌식'을 이용하여 수사가 진행된다. 이 때 삭제된 데이터의 복구가 중요하며 관련 연구가 많이 진행되고 있지만 삭제 방법보다 복구 방안에 초점을 두고 진행하는 경우가 많다. 따라서 본 논문에서는 실제 수사에서 데이터 복구 실패한 사례를 통해 기존 연구된 삭제된 레코드 복구 기법 방식을 실험한다.

1. 서론

데이터베이스는 대용량의 데이터를 효율적으로 관리할 수 있는 장점을 가지고 있기 때문에 다양한 분야에서 사용되고 있다. 축적된 데이터를 활용하는 분야가 점점 늘어나고 있으며 데이터의 중요성 또한 높아지고 있다. 데이터가 중요해지고 있는 만큼 기업, 정부의 핵심 정보를 취득, 조작하려는 관련 범죄가 늘어나고 있다. 이러한 범죄는 PC, 스마트폰 등 다지털 기기의 정보를 수집, 복구, 분석해 활용하는 '디지털포렌식'을 이용하여 수사가 진행된다.

조희팔 다단계 사건은 디지털포렌식을 이용하여 수사가 진행된 사건 중 하나이다. 당시 주범들은 데이터파일을 여러번 덮어쓰기, 기술적 삭제 등을 이용하였고 수사관들은 해당 데이터를 복구하는데 어려움이 많았다고 한다.

포렌식 관점에서 데이터를 복구하는 기법에 대해 다양한 연구가 진행되고 있는데 의도적으로 데이터를 삭제, 조작하는 경우를 고려해야 한다. RDBMS, No SQL 등 다양한 데이터베이스에서 복구 기법에 관한 연구가 진행 중이며 이 논문에서는 데이터베이스에서 삭제된 레코드 복구 기법의 한계를 논하고자 한다.

데이터베이스에서 삭제 된 레코드를 복구하는 기법은 크게 2가지가 있다. 첫째는 트랜잭션 로그를 분석하여 작업 과정을 추적하여 데이터베이스를 재구성 하는 기법이다. 둘째는 레코드가 저장되는 데이터베이스 파일의 구조를 분석하여 비할당 영역에 잔존하는 데이터를 복구하는 기법이다. 이는 트랜잭션 로그 파일이 없더라도 복구할 수 있다는 장점을 가지고 있다.

2. 관련 연구

Theo Haerder는 다양한 종류의 데이터베이스들에 적용할 수 있는 범용적인 삭제된 레코드 복원 방법을 제시하였으나, 이는 트랜잭션 로그에 기반한 방법론으로 복원할 수 있는 데이터가 트랜잭션 로그 내에 존재하는 경우로 한정되며, 트랜잭션 로그가 존재하지 않는 시에는 적용할 수 없다.

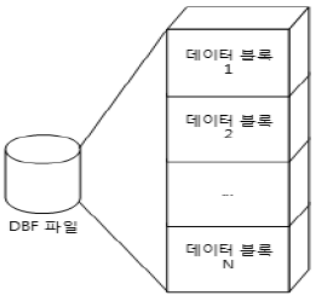
그리고 대부분은 데이터베이스 제품군 별로 삭제된 레코드 복원 방법에 대한 연구가 진행되어 왔다. SQLite 데이터베이스의 경우, S.Jeon은 SQLite 데이터베이스 파일의 구조적인 특성을 이용하여 비할당 영역으로부터 삭제된 레코드를 복원하는 방법에 대해서 제시하였다. 또한, Oracle, MySQL, Microsoft SQL Server, MongoDB 등 삭제된 레코드를 비할당 영역으로부터 복구하는 기법이 제시되었다.

그러나 대부분의 연구에서 진행 한 실험은 단순 삭제 이벤트를 통해 진행되었다. 포렌식 관점에서 데이터를 복구하는 이유는 데이터를 의도적으로 삭제, 폐기 된 핵심 정보를 확인하려는데 있다. 따라서 여러 가지 방법을 통해 삭제를 하여 위 연구에서 제시하는 복구 기법이 적용될 수 있는지 확인하고자 한다.

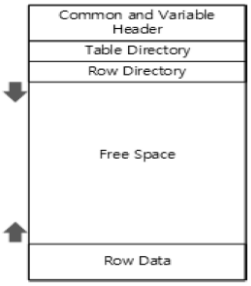
3. 데이터베이스 파일 구조

3-1. Oracle 파일 구조

Oracle 테이블스페이스는 TEMP, USER, SYSAUX, SYSTEM, EXAMPLE, UNDOTBS가 있으며, SYSTEM에서 테이블과 레코드들을 저장하고 있다.



(그림 1) DBF 파일 구성

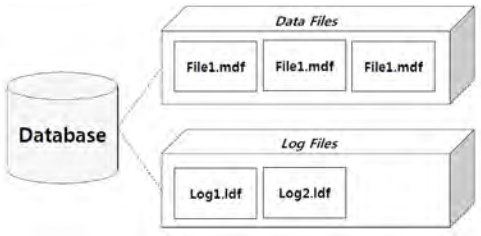


(그림 2) 데이터 블록 구조

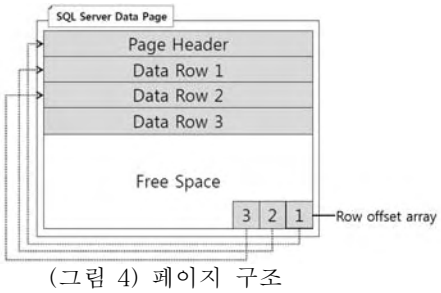
테이블스페이스는 (그림 1)처럼 데이터 블록으로 구성되어 있으며 (그림 2) 데이터 블록은 데이터를 읽거나 쓸 때 사용되는 조작 단위이다.

3-2. Microsoft SQL Server 파일 구조

Microsoft SQL Server 데이터베이스에서 관리하는 데이터 물리 원본 파일은 데이터파일과 로그파일로 구분된다. 데이터 파일은 Database Object 들을 저장하고 있는 파일을 말하며 (*.mdf)로 생성된다. 로그파일은 DML (Database Manipulation Language)과 DDL (Database Definition Language)의 동작(트랜잭션 단위)을 기록하는 파일이며 (*.ldf)로 저장된다.



(그림 3) 데이터베이스 구성



(그림 4) 페이지 구조

페이지는 데이터 기본 저장 단위이며 Data Row의 집합으로 이루어진 8192 Byte 크기의 구조체이다. 기본적으로 Page Header, Data Rows, Rows Offset Array로 구성되어 있다.

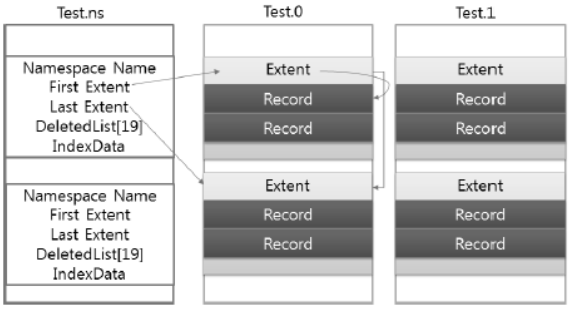
각 페이지는 96Byte 헤더로 시작되며 해당 페이지에 대한 시스템 정보를 저장하고 있다.

Data Row는 헤더 다음부터 시작하는 실제 데이터가 저장되는 영역으로 한 페이지에서 한 행의 최대 크기는 8060Byte이다. 여기서 텍스트나 이미지, 8060Byte를 초과하는 가변 길이 컬럼은 ROW_OVERFLOW_DATA 할당 영역에 있는 페이지에 저장되고 원래 페이지에는 24Byte의 포인터 정보만 저장하게 된다.

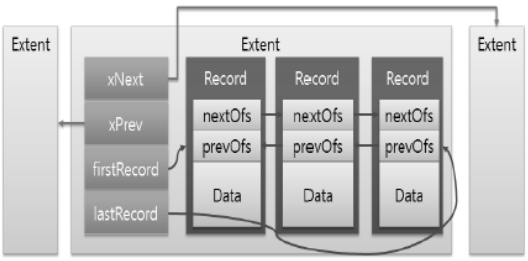
Offset Array는 한 페이지에 저장되어 있는 테이블 행의 첫 번째 Byte가 페이지 시작으로부터 얼마나 떨어져 있는지 정보를 저장하는 구간이다. 페이지 끝에서부터 2Byte씩 배열 형태로 조장되어 있다.

3-3. MongoDB 파일 구조

MongoDB 데이터 파일은 DB의 메타정보를 저장하는 파일과 데이터를 저장하는 파일로 구분된다. 데이터 구조는 크게 Namespace-Details, Extent, Record로 구성된다. Name-spaceDetails 구조체는 “DB이름. Collections 이름” 형태의 Namespace가 사용하는 Extent의 위치, 삭제된 레코드 리스트, 레코드 크기, 개수, 인덱스 관련정보 등 메타정보를 저장하고 있다. Namespace 파일의 NamespaceDetails 구조체를 통해 Extent와 Record를 따라가면서 파싱하면 Namespace 파일과 데이터 파일에서 데이터를 추출할 수 있다. (그림 5)는 MongoDB의 전체적인 데이터 구조를 보여준다.



(그림 5) 데이터파일 구조



(그림 6) 익스텐트와 레코드 구조

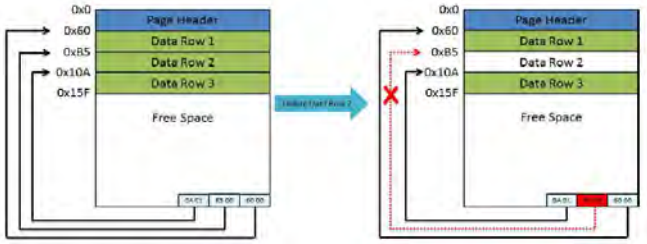
Extent는 연속된 블록의 집합으로, 같은 Namespace에 속한 Extent는 이중 링크드리스트로 연결되어 있고 각 Extent에는 Record가 속해 있다. Record는 BSON Document나 B-tree 형태의 데이터를 저장하고 있으며 관

련된 Record끼리 이중 링크드리스트로 연결되어 있다. (그림 6)은 Extent와 Record의 구조를 보여준다.

4. 실험

데이터베이스 파일 구조는 전체적으로 비슷한 구조를 가지며 본 논문에서는 Microsoft SQL Server로 복구 기법 설명 및 실험을 진행했다.

레코드를 삭제 한 경우 페이지에서 데이터는 지워지지 않고 해당 Data Row를 가리키고 있던 Row Offset 부분만 0으로 초기화된다. 따라서 삭제를 했지만 비할당 영역에는 데이터가 존재하며 해당 영역에 새로운 데이터가 덮어쓰기 전까지 계속 존재한다.



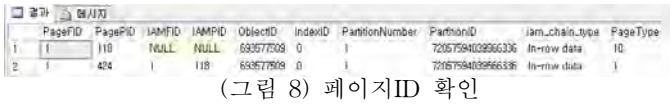
(그림 7) 레코드 삭제 시 페이지 구조 변화

각각의 레코드 내에 길이 정보를 얻는다. 따라서 해당 길이와 비할당 영역의 크기와 비교함으로써 유효한 레코드인지 여부를 확인한다. 계산된 레코드의 길이가 비할당 영역의 크기와 일치한다면 데이터 추출 과정을 진행하여 복구한다.

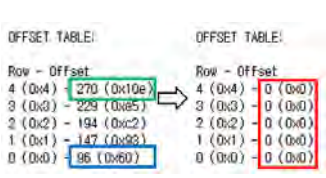
4-1. 레코드 삭제 실험

실험은 테스트 테이블을 생성하여 레코드 삭제 후 INSERT를 여러번 반복 방식으로 진행했다.

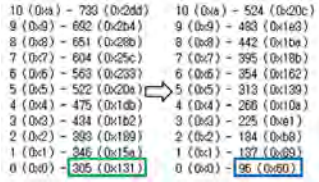
TEST 테이블을 생성하여 페이지 ID 424를 확인한다.



(그림 8) 페이지ID 확인

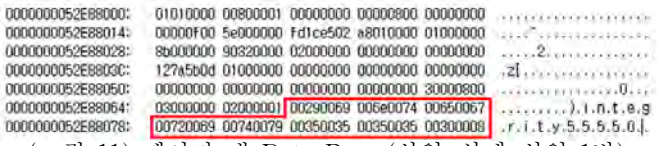


(그림 9) Offset table1-2

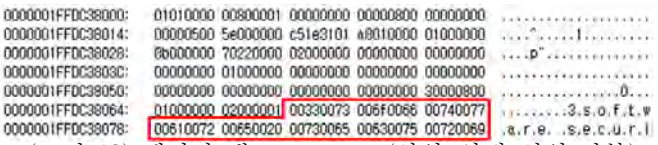


(그림 10) Offset table3-4

(그림 9) Offset table1은 처음 삽입 된 레코드와 Offset table2는 삭제 된 모습을 보여준다. 삭제된 후 0으로 초기화된 데이터를 볼 수 있다. (그림 10) Offset table3은 테이블 레코드 삭제 후 INSERT, Offset table4는 이를 반복한 모습이다. 첫 번째 0 Row 에는 305(0x131) 값이 들어가 있으며 이는 (그림 9) 첫 번째 Offset 가장 상위 270 (0x10e) 보다 큰 값이다. 그러나 삭제와 삽입 반복 후 0 Row를 보면 96 (0x60) 값이 들어가 있는데 이는 (그림 9) 첫 번째 0 Row 데이터와 같다. 즉 처음 삽입했던 Data Row 위치에 데이터가 덮어쓰기 된 것을 볼 수 있다.



(그림 11) 페이지 내 Data Row (삽입-삭제-삽입 1번)



(그림 12) 페이지 내 Data Row (삽입-삭제-삽입 반복)

(그림 11) 데이터는 삽입-삭제-삽입 과정을 거쳐도 그대로 남아있다. 그러나 위 과정을 여러번 반복 후 (그림 12)와 같이 데이터가 덮어쓰기 된 것을 볼 수 있다.

초기 레코드 삭제 시 데이터는 지워지지 않고 비할당 (Free Space) 영역에 존재한다. 그러나 해당 과정을 반복할 경우 비할당 영역에 있는 데이터에 다른 데이터들로 덮어쓰기 된 것을 확인 할 수 있다.

5. 결론

데이터베이스는 계속 발전하는 환경에 따라 용도와 중요성이 더욱 커질 것이다. 관련 범죄 시 데이터베이스는 증거로써 중요한 역할을 하게 될 것이며 데이터의 조작유무에 따라 채택이 결정될 것이다. 현재 데이터베이스의 삭제된 레코드 복원 등 다양한 연구가 진행되고 있다. 범죄자들은 기술적으로 데이터를 삭제하고 점점 지능화 되고 있다는 점에서 관련 실험들의 방식을 다양화 할 필요가 있다.

데이터베이스별 삭제된 레코드 복구 관련 논문들을 찾아보니 대부분 단순 삭제 후 복구방안에 중점을 둔 연구였다. 실제 수사에서 데이터 복구에 실패한 사례를 통해 이 논문에서는 삽입-삭제-삽입을 반복하는 등 다양한 방법을 통해 실험을 진행한 결과 기존 연구와는 다른 결과물을 도출 할 수 있었다.

앞으로 관련 연구에서는 삭제 방법과 복구 방안 모두 강구하여 진행하는 동시에 레코드 삭제, 변조를 다양하게 시도하여 연구해야 할 것이다.

참고문헌

[1] Theo Haerder, Andreas Reuter, “Principles of Transaction-Oriented Database Recovery”, Computing Surveys, Vol.15, No.4, Dec.1983

[2] Sangjun.J, Jewan.B, Keunduck.B, Sangjin.L, “A recovery method of deleted record for SQLite database”, Personal and Ubiquitous Computing, Vol.16 ,No6, Aug.2012
[3] Jong-Hyun Choi, DooWon Jeong, Sangjin Lee, "The method of recovery for deleted record in Oracle Databas e", Korea Institute Of Information Security And Cryptol ogy, Vol.23, No.5, Oct.2013
[4] SooYoung Park, "A Research for Record Recovery Method in Database", Thesis for the Degree of Master, Dec.2013
[5] Ryu Gi Hwan, "A Study for Recovering Records of Microsoft SQL Server's Database", Thesis for the Degr ee of Master, Dec.2014
[5] Jong-Seong Yoon, Doo-Won Jung, Chul-hoon Kang, Sang-Jin Lee, "Digital Forensic Investigation of Mongo DB", Journal of the Korea Institute of Information Secu rity & Cryptology, Vol.24, No.1, Feb. 2014.
[6] Microsoft SQL Server, https://en.wikipedia.org/wiki/ icrosoft_SQL_Server
[7] Database, https://en.wikipedia.org/wiki/Database
[8] 데이터베이스, http://forensic.korea.ac.kr/DFWIKI/ind ex.php/%EB%8D%B0%EC%9D%B4%ED%84%B0%EB% B2%A0%EC%9D%B4%EC%8A%A4
[9] 디지털포렌식 활용 수사사례, https://cpuu.postype.com /post/220456