데이터베이스 포렌식에서 칼럼 정보 없이 삭제된 레코드 복구를 위한 분석

김선경*, 박지수**, 손진곤*+
*한국방송통신대학교 정보과학과
**충남대학교 SW중심대학사업단
skkimic@knou.ac.kr

Record extraction from database pages without columns information

Sunkyung Kim*, Ji Su Park**, Jin Gon Shon*
*Dept of Computer Science, Korea National Open University
**Worldwide Innovative SW Experts, Chungnam National University

요 약

파일시스템 포렌식으로 삭제된 파일을 복구할 수 있지만 데이터베이스에서 이용한 데이터 파일의 경우 레코드는 복구할 수 없다. 따라서 데이터베이스 포렌식이 연구되고 있다. 데이터베이스 포렌식은 각 데이터베이스 고유 저장 특성을 이용하여 데이터파일에서 삭제된 레코드를 추출하는 기법이다. 그러나 칼럼 정보를 얻지 못할 경우 기존 데이터베이스 포렌식으로는 레코드 복구가 불가능하다. 데이터 파일의 구성단위인 페이지를 읽어 칼럼구조를 유추하면 레코드를 복원할 가능성이 있다. 본 논문은 칼럼 정보 없이 데이터 파일에 레코드가 저장된 형태를 분석하여 삭제된 레코드를 포함한 파일에 잔존하는 레코드를 복구를 위한 분석과 실험을 하였다.

1. 서론

현재 대부분의 컴퓨터 시스템은 데이터베이스를 이용한다. 데이터베이스 내에는 각종 주요 정보가 존재한다. 따라서 범죄 행위가 의심되는 조직이나 개인을 수사할 때 데이터베이스 조사는 수사기관이 반드시 수행해야 하는 절차다.

수사를 받는 대상은 증거 인멸 목적으로 수사가 시작되기 전에 데이터베이스 내의 레코드를 훼손하는 것이 가능하다. 따라서 수사기관은 데이터베이스 시스템을 압수했을 때 정상적으로 추출되는 데이터를 조사하는 것도 중요하지만 훼손된 레코드를 추출하는 것도 중요하다. 삭제된 레코드의 경우 조직에서 숨기고자 하는 데이터일 가능성이 높기 때문에 수사기관의 관점에서 더욱 의미 있는 물리적 증거가 된다[1].

데이터베이스는 물리적으로 데이터파일을 포함한 일반 파일에 메타 데이터와 실 데이터를 저장한다[2]. DBMS는 레코드를 삭제하더라도 성능을 위하여 레코드의 모든 값을 의미 없는 값으로 변경하지 않고 삭제 여부만을 표시하는 것이 일반적이다. 따라서 데이터 파일을 직접 분석하면 DBMS에서 삭제된 레코드라도 원래의 값을 추출하는 것이 가능하다.

많은 연구에서 모든 데이터 파일을 획득할 수 있고 그곳으로부터 칼럼의 구조 정보를 얻을 수 있다는 전제하에 복구 기법을 제안하고 있다. 하지만, 수사현장에서는 일부의 데이터 파일만 획득하여 칼럼의 구조를 직접적으로 파악이 불가능할 수 있다. 본 연구는 실제로 데이터가 저장되어 있는 일부 데이터파일만을 획득했을 경우에도 칼럼 구조를 유추하여 삭제된 레코드를 포함한 데이터를 복구를 이한 분석을 하고 실험을 통하여 복구 가능성을 보여준다. 실험은 전 세계적으로 가장 많이 사용되는 Oracle 데이터베이스, MySQL InnoDB 데이터베이스, 스마트폰에서 사용되는 SQLite를 대상으로 실험하였다[3].

2. 관련 연구

2.1 데이터베이스 포렌식

데이터베이스 포렌식 초기 연구에서는 모바일 기기에서 개인이 사용하는 SQLite 데이터베이스 데이터 파일에서 삭제된 레코드를 추출하는 기법이 주로 연구되었다. 전상준[4]은 SQLite 데이터베이스의 비할당 영역에서 잔존하는 삭제된 레코드를 복구하는 기법을 제시하고 대표적인 모바일 기기인 아이폰4에서 삭제된 문자 메시지 데이터 복구를 실험하였다. 이규원[5]은 SQLite 데이터베이스의 삭제된 오버플로우 데이터 복구 기법을 소개하고 대표적인 문자메시지 앱(App)인 카카오톡에서 삭제된 메시지를 복구하는 실험을 하였다. 이후 기업 등에서 다수의 사용자가 이용하는 데이터베이스의 데이터 파일에서 삭제된 레코드를 추출하는 기법이 연구되었다. 최종현[6]은

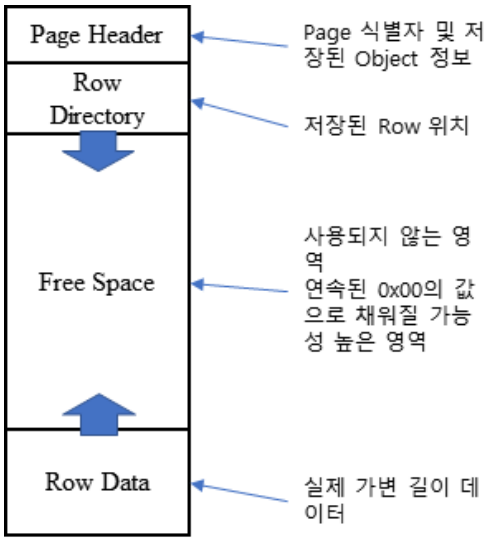
+ 교신저자

Oracle 데이터베이스의 삭제된 레코드 복구 기법을 소개하고 실험하였다. Jang[7]은 MySQL InnoDB 스토리지 엔진의 구조를 분석하여 삭제된 데이터를 레코드 단위로 복구하는 기법을 제안하고 제작한 도구를 활용하여 이를 검증했다. James Wagner[8]는 데이터 파일에서 삭제된 데이터를 복구하는 공통적인 특징을 분석하고 8개 DBMS에 대해서 각각의 개별적인 특징을 분석하고 데이터 복구 도구를 구현하였다.

선행 연구는 데이터베이스 파일에서 칼럼의 구조를 직접적으로 획득 가능하다는 전제하에 데이터베이스 포렌식 기법이 연구되었다. 그러나 일부 파일만을 획득한 경우 연구된 포렌식 기법 적용이 어렵다는 단점이 있다.

2.2 데이터베이스 페이지 구조

일반적인 데이터베이스 페이지 구조는 그림 1과 같은 형태를 가진다. 페이지 앞쪽에서 고정된 크기의 페이지 헤더가 위치하며 페이지를 식별할 수 있는 정보와 페이지가 가지고 있는 데이터베이스 객체(Table이나 Index 등) 정보를 가지고 있다.

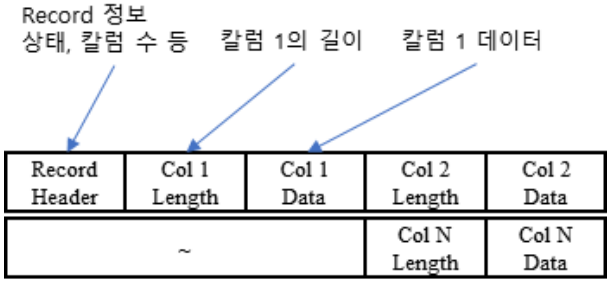


(그림 1) 페이지 구조

데이터베이스는 공간을 효율적으로 관리하기 위하여 슬롯페이지 구조를 이용한다. 가변길이 레코드는 페이지 가장 끝에서부터 입력되는 순으로 위쪽으로 채워져 나가고 각 Row의 시작 주소를 알기 위하여 Row 주소는 페이지 헤더 바로 뒤에서부터 아래로 빈 공간을 채워나가는 구조를 가진다[9]. DBMS 종류에 따라서 Row 위치를 저장하는 Row Directory가 아래서부터 채워지고 실제 데이터인 Row Data는 페이지 헤더 바로 뒤에서부터 채워지는 반대의 구조를 같기도 한다. 본 논문 실험 대상인 Oracle과 SQLite 데이터베이스는 Row Data가 바닥에서 올라오는 구조이고 MySQL InnoDB는 페이지 헤더 다음부터 아래로 채워지는 구조를 가지고 있다.

2.3 레코드 저장 구조

그림 2와 같이 레코드의 시작을 알리고 레코드가 가진 칼럼 수 등 레코드 메타 데이터를 가지고 있는 Record Header가 가장 앞에 위치한다. 그리고 칼럼의 길이와 칼럼 실제 데이터가 차례로 저장되어 있다. DBMS 종류에 따라 Record Header 구조와 칼럼 데이터의 구조가 차이가 있다. 그러나 문자 데이터의 경우 데이터베이스가 사용하는 문자집합 코드 값이 연속적으로 저장되어 있는 것이 일반적으로 관찰된다. 그리고 칼럼의 길이 대신에 칼럼 구분자를 이용하여 칼럼의 끝과 시작을 표시하기도 한다.



(그림 2) Record 저장 구조

3. 데이터 파일에서 삭제된 레코드 분석

Page 크기와 DBMS의 종류를 가정하고 Free Space에서부터 Row Data가 있는 방향으로 파일을 읽으며 데이터베이스가 일반적으로 가지고 있는 레코드 저장구조에 일치하는지를 검사해 나가며 레코드를 추출하는 것이 가능하다.

Oracle 레코드 헤더는 정상적인 경우 0x2C값으로 시작하며 삭제된 Record의 경우 0x3C로 시작한다. 또한 헤더에 칼럼 수가 나온다. 그 뒤에 칼럼의 길이 및 칼럼 데이터가 따라 나온다. 따라서 Row Data 영역을 읽어가면서 0x2C값이 발견될 경우 레코드 시작으로 추정하고 바로 뒤에 나오는 길이 정보를 이용하여 데이터를 읽을 수 있다. 칼럼에 대한 메타데이터가 없으므로 칼럼 데이터 형(形)은 데이터 길이와 패턴을 이용하여 추정한다. 칼럼 길이가 7이고 첫 데이터가 120이거나 119인 경우 DATE 형으로 예상할 수 있다. 120인 경우는 2000년대를 119인 경우 1900년대를 의미한다. 첫 데이터가 0xC1에서 0xCB 사이의 값인 경우 Number 데이터 형의 예상된다. 하위 4bit가 Number 데이터 형의 유효자릿수/2의 올림 값을 의미한다. 칼럼 길이 데이터 이후 데이터가 모두 유효한 문자 영역(0x20~0x7E)인 경우 문자열일 가능성이 있다. 레코드가 각 컬러의 데이터의 패턴을 참조하여 각 칼럼의 형을 유추할 수 있다.

MySQL InnoDB 데이터파일을 분석할 결과 Row Data가 페이지 헤더 뒤쪽부터 아래쪽으로 쌓이는 것을 확인했다. 데이터가 있는 페이지의 헤더는 항상 일정한 문자열

로 끝나는 것을 확인 했다. 따라서 페이지 처음부터 스캔을 하여 일정한 문자를 만나면 페이지 헤더의 종료점으로 추정할 수 있다. InnoDB의 레코드 구조는 그림 3과 같다 [10].

(그림 3) MySQL InnoDB Record Header 구조

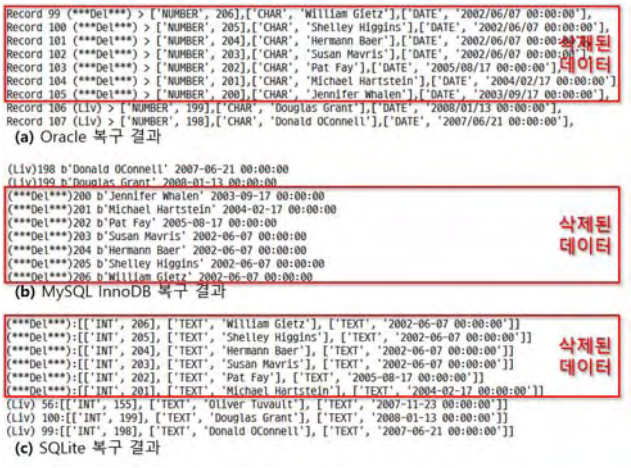
Extra Bytes의 위치를 찾아내면 다음 레코드의 위치를 찾을 수 있으므로 모든 Page에서 레코드별로 분리할 수 있다. Extra Bytes중 레코드 시퀀스가 13bit 있다. 레코드 시퀀스 값은 2부터 시작한다. 따라서 비트 패턴 매칭을 하여 Extra Bytes의 위치를 추정할 수 있다. 실제 데이터 중 문자열은 오라클과 동일하게 유효한 문자 코드 값 반복 출현으로 파악하며 숫자는 최상위 비트가 1로 시작하는 바이트로 시작하는 것이 특징이다. 날짜와 시간을 저장하는 DATETIME 형의 경우 'YYYYMMDDHHMISS'로 표현된 것을 10진수로 저장한다. 이러한 패턴을 나타는 것을 판별하여 실제 데이터와 자료형을 분석할 수 있다. 레코드가 삭제되는 경우 Info Flags 3번째 bit가 1로 처리되는 것을 이용하여 삭제된 레코드를 복구할 수 있다.

SQLite페이지는 Internal과 Leaf 페이지로 구분되며 Leaf 페이지에 모든 데이터가 저장되어 있다. MySQL InnoDB는 가변길이 정수 값 구조라는 고유한 특성을 가진다. 각 바이트의 하위 7bit를 정수 값으로 사용하며 최상의 Bit가 1이면 값이 계속되는 것이며 0이거나 9번째 값이면 정수 값이 끝나는 것이다. Row Data에 레코드의 길이 속성에 각 필드의 길이와 필드 형 정보를 가지고 있다. 따라서 이러한 특성을 이용하여 레코드를 해석하면 데이터를 추출할 수 있다. 삭제된 데이터는 RowID값이 0x00으로 변경된다.

4. 실험

Oracle의 NUMBER, DATE, VARCHAR2 형에 대해서 앞에서 설명한 패턴을 참조하여 유추한 후 데이터를 해석하면 각 칼럼의 데이터를 추출할 수 있다. 레코드 헤더가 0x3C인 경우 삭제된 레코드임을 확인할 수 있다. 오라클 테이블의 일부 데이터를 삭제 후 테이블이 위치한 영역의 페이지를 분석해서 그림 4 (a)와 같이 정상적인 데이터와 삭제된 데이터까지 얻었다.

MySQL InnoDB 포렌식 실험을 위하여 Windows 환경에 MySQL 5.7 준비하고 테스트 테이블을 생성하였다. 테이블에 넣었던 Row를 삭제 후 Hex Editor로 파일을 열어서 검색한 결과 MySQL 5.7에서는 삭제된 Row의 모든 Row 데이터가 0x00으로 초기화된 것을 확인했다. 결국 MySQL 5.7 버전에서는 포렌식이 불가능했다. 그러나 현장에서는 5.7 하위 버전을 사용할 가능성도 있으므로 포



(그림 4) 데이터 복구 결과

렌식이 가능한 하위 버전에서의 실험이 의미가 있다. MySQL 5.5 버전에서 동일한 실험을 한 결과 삭제된 Row 데이터가 남아 있음을 확인했다. 3장에서 제시한 MySQL InnoDB 레코드 추출 기법을 실험한 결과 그림 4 (b)와 같이 삭제된 데이터가 복구됨을 확인했다.

SQLite에 INTEGER형 칼럼과 TEXT형 칼럼 두 개를 만들었다. SQLite는 날짜와 시간을 저장하는 DATE형은 지원하지 않는다. 따라서 TEXT형에 YYYY-MM-DD HH:MI:SS 형태의 문자열로 날짜시간을 데이터를 만들어 넣었다. 그런 후 일부 데이터를 삭제하고 3장에서 분석한 레코드를 토대로 그림 4 (c)와 같은 결과를 없었다.

5. 결론

최근 수사현장에서 증거 수집을 위한 디지털 포렌식이 일반화되었다. 주로 저장매체에서 삭제된 파일을 복구하여 증거를 수집한다. 그러나 데이터베이스의 경우 파일 단위의 복구로는 충분하지 않으며 레코드 단위의 복구가 필요하다. 이런 이유로 많은 연구에서 DBMS별 고유 특성을 이용하여 레코드를 복구하는 기법을 제안했다. 이런 기법을 적용하기 위해서는 메타데이터를 파악하기 위해서 우선 딕셔너리(Dictionary) 테이블이 분석되어야 했다. 이런 것이 실패할 경우 데이터가 들어 있는 데이터 파일이 있음에도 불구하고 데이터 추출이 불가능했다. 본 연구의 동기는 데이터 파일만을 가지고 테이블에 대한 메타데이터 없이 레코드의 추출을 위한 분석을 하고 가능성을 확인하기 위하여 실험하였다.

페이지 크기와 DBMS의 종류를 추정 후 데이터 저장되는 일반적인 형태를 추정하여 각 자료형을 추정하고 데이터 추출이 가능했다. 대상이 되었던 DBMS는 윈도우 환경에서 Oracle, MySQL InnoDB, SQLite로 하였으며 칼럼 형은 가장 많이 사용되는 문자열, 시간, 숫자 형을 분석하였다.

본 연구에서는 페이지 크기와 DBMS의 종류를 추정 후 페이지 분석을 하였다. 그러나 추정이 틀릴 경우 원하는 결과를 얻을 수 없는 단점이 있다. 추가연구에서 페이지 크기와 DBMS 종류를 데이터파일 분석하여 알아내는 기법 개발이 필요하다.

참고문헌

[1] 대검찰청, "https://www.spo.go.kr/spo/major/forensics /act/ forensics10.jsp", 대검찰청 홈페이지 디지털 수사지원, 2017. 9. 17 검색

[2] James Wangner, Alexander Rasin, Tanu Malik, Karen Heart, Hugo Jehle, "Database Forensic Analysis with DBCarver", 8th Biennial Conference on Innovative Data Systems Research (CIDR '17), 2017, 1

[3] DB-ENGINES, "http://db-engines.com/en/ranking", 2017.05.28. 검색

[4] 전상준, 변근덕, 방제완, 이근기, 이상진, "SQLite 데이터베이스의 비 할당 영역에 잔존하는 삭제된 레코드 복구 기법", 정보보호학회논문지 21(3), pp.143-154, 2011.6,

[5] 이규원, 양스제, 황현욱, 김기범, 장태주, 손기욱, "SQLite 데이터베이스의 삭제된 오버플로우 데이터 복구 기법", 한국정보기술확회논문지 10(11), pp.143-153, 2012.11,

[6] 최종현, 정두원, 이상진, "Oracle 데이터베이스의 삭제된 레코드 복구 기법", 정보보호학회논문지 23(5), pp.947-955, 2013.10

[7] Jeewon Jan, Doowon Jeoung, Sang Jin "The Recovery Method for MySQL InnoDB Using Feature of IBD Structure", KIPS Tr. Comp. and COmm Sys. Vol.6 No.2. pp.59-66, 2016.11

[8] James Wagner, Alexander Rasin, Jonathan Grier, "Database forensic analysis through internal structure carving", Digital Investigation 14, S106-S115, 2015.5

[9] Abraham Silberschatz, Henry F. Korth, S.Sudarshan "Database System Concepts 6th Edition", McGraw-Hill Education, pp.456-457, 2010.1

[10] The physical structure of records in InnoDB, "https://blog.jcole.us/2013/01/10/the-physical-structure-of-records-in-innodb/", 2017. 9. 17 검색