情報保護學會論文誌
第 21 卷 第 3 號, 2011. 6

SQLite 데이터베이스의 비 할당 영역에 잔존하는 삭제된 레코드 복구 기법*

전 상 준†, 변 근 덕, 방 제 완, 이 근 기, 이 상 진‡
고려대학교 정보보호연구원

The Method of Recovery for Deleted Record in the Unallocated Space of SQLite Database*

SangJun Jeon†, Jewan Bang, KeunDuck Byun, GuenGi Lee, SangJin Lee‡
Center for the Information Security Technologies

요 약

SQLite는 소형 데이터베이스로 임베디드 기기와 로컬 응용프로그램에서 주로 사용된다. 최근에는 스마트폰을 비롯한 휴대용 디지털 기기의 보급이 확대됨에 따라 SQLite의 사용이 더욱 증가하고 있다. 따라서 포렌식 수사 과정에서 수집된 디지털 증거에 SQLite 데이터베이스 파일이 포함되어 있을 가능성이 많으며, 용의자가 의도적으로 민감한 데이터를 삭제할 가능성이 있기 때문에, 조사시 SQLite의 삭제된 레코드를 최대한 복구할 필요가 있다. 이 논문은 SQLite의 데이터 관리 규칙과 삭제된 데이터의 구조를 분석하였고, 이를 토대로 삭제된 후 덮어 쓰여지지 않은 레코드의 복구 방법을 제시하였다. 또한 SQLite를 사용하는 대표적인 소프트웨어를 조사하여 삭제된 데이터의 복원 가능성을 확인하였다.

ABSTRACT

SQLite is a small sized database engine largely used in embedded devices and local application software. The availability of portable devices, such as smartphones, has been extended over the recent years and has contributed to growing adaptation of SQLite. This implies a high likelihood of digital evidences acquired during forensic investigations to include SQLite database files. Where intentional deletion of sensitive data can be made by a suspect, forensic investigators need to recover deleted records in SQLite at the best possible. This study analyzes data management rules used by SQLite and the structure of deleted data in the system, and in turn suggests a recovery Tool of deleted data. Further, the study examines major SQLite suited software as it validates feasible possibility of deleted data recovery.

Keywords: SQLite, Database, Recovery, Record, Structure, tool

I. 서 론

대부분의 데이터가 디지털 형태로 변환되어 저장됨에 따라 방대한 양의 데이터를 효율적으로 관리하기 위해 데이터베이스의 사용이 꾸준히 증가하고 있다. 데이터베이스에 저장되는 정보는 시스템 운용에 필요한 핵심 정보인 경우가 많으며 포렌식 조사시 데이터베이스에서 의미 있는 정보를 찾게 될 가능성이 많아진다. 또한 데이터베이스에서 효율적인 저장 공간의 관리나 최신 데이터 유지를 위해 삭제 행위가 빈번하

접수일(2010년 6월 19일), 게재확정일(2010년 10월 25일)
* 본 연구는 한국연구재단을 통해 교육과학기술부의 바이오연구개발사업으로부터 지원받아 수행되었습니다.(20100020634)
† 주저자, heros86@korea.ac.kr
‡ 교신저자, sangjin@korea.ac.kr

게 발생한다는 것을 고려할 때 데이터베이스에서 정상적으로 남아있는 데이터를 추출하는 것만큼 삭제된 데이터에 대한 정보를 획득하는 것도 중요하다.

SQLite는 ANSI-C를 기반으로 제작된 오픈소스 데이터베이스 소프트웨어로, 작고 빠른 특징 때문에 간단하지만 많은 양의 정보를 체계적으로 저장할 필요가 있는 응용프로그램이나 임베디드 기기에 탑재된 소프트웨어에서 주로 사용되고 있다. SQLite는 현재도 꾸준히 업데이트 되고 있으며 많은 개발자들에 의해 선호되므로 사용분야가 앞으로도 늘어날 것으로 보인다. 따라서 포렌식 조사에서 임베디드 기기의 증거를 수집할 경우 상당량의 데이터가 SQLite 데이터베이스 파일에 저장되어 있을 가능성이 많다. 또한 대부분의 임베디드 장치는 작은 저장 공간을 사용하므로 다른 환경에 있는 데이터베이스에 비해 삭제가 빈번하게 발생된다. 따라서 SQLite 데이터베이스에서 삭제된 레코드를 복원하는 것은 상당히 중요하다.

II. 관련 연구

SQLite를 사용하는 응용프로그램은 데이터베이스로 쿼리를 전송하는 과정에서 "〈데이터베이스 파일명〉.〈확장자〉-journal" 이라는 파일 이름을 가진 저널 파일을 생성하여 쿼리 전송이 종료되기 이전의 레코드 입출력 과정을 저장해 놓는다. 이러한 저널 파일을 통해 파이어폭스3의 삭제된 레코드를 복구하는 연구가 M. T. Pereira에 의해 진행된 바 있다[1]. 또한 기존의 데이터베이스 레코드 복구 연구에서는 레코드를 복구할 때 SQLite의 저널파일과 같이 데이터베이스의 쿼리 수행 내용을 기록한 트랜잭션을 토대로 복구가 가능한 영역을 복구하는 방법을 제시해왔다.

본 논문에서는 Pereira의 연구와 마찬가지로 SQLite에서 삭제된 레코드를 복구하는 것을 목적으로 하지만, 저널 파일이 아닌 실제 데이터 파일에 접근하여 잔존하는 삭제 데이터를 복구하는 범용적으로 활용 가능한 레코드 복구 기법을 제안한다.

III. SQLite

3.1 SQLite

SQLite는 오픈소스 데이터베이스 소프트웨어로 가볍고 빠른 특성 때문에 간단한 정보를 체계적으로 저장할 필요가 있는 응용프로그램이나, 임베디드 장치에 탑재되는 소프트웨어에서 많이 사용되고 있다. 대부분의 데이터베이스와 다르게 SQLite는 주로 로컬 전용 데이터베이스로 사용되면서, 데이터베이스 사용에 대한 결과는 하나의 파일에 모두 저장된다. 응용프로그램의 구성품으로 탑재된 SQLite 엔진은 데이터베이스 파일을 직접 읽고 쓰면서 테이블, 인덱스, 트리거, 뷰를 갖춘 하나의 파일로 관리한다. 작은 크기가 특징인 SQLite는 모든 기능을 사용 가능하도록 설정할 경우 라이브러리 크기가 250KB정도 되며, 필수 기능 요소만 사용 가능하도록 할 경우 180KB 정도의 크기를 갖는다. 또한 최소 스택 공간인 16KB 스택 공간에서 사용하도록 만들 수 있으며 100KB정도의 매우 작은 힙 크기를 사용한다. 따라서 휴대전화, PDA, MP3 플레이어와 같이 가용할 수 있는 메모리가 적은 환경에서 데이터베이스 엔진을 구성할 수 있다[2].

일반적으로 신뢰성이 높다고 알려진 SQLite는 표준 SQL(ANSI/ISO SQL)을 대부분 지원하며, 지원하지 않는 리스트는 홈페이지[3]에 정리되어 있다. 또한 트랜잭션은 전원 이상과 같은 예기치 못한 시스템 오류에도 ACID(Atomic, Consistent, Isolate, Durable)특성을 보증한다[2].

ANSI-C로 짜여진 SQLite는 TCL을 비롯하여 다양한 언어에 대한 바인딩을 포함하고 있으며 99% 이상 테스트가 진행된 외부 의존성이 없는 코드 이다. 또한 리눅스, 유닉스, Mac OS X, OS/2, Win32를 지원하고 서로 다른 운영체제, 플랫폼, 아키텍처로의 변환이 용이하다[2].

3.2 사용 분야

현재 많은 응용프로그램이 프로세스 과정에서 생성되는 정보를 SQLite를 사용하여 저장한다. 대표적으로는 사파리, 아이폰 OS, 파이어폭스3, 크롬, 안드로이드, 스카이프가 있다[5]. 각 응용프로그램에서 SQLite 파일을 저장하는 경로와 저장하는 내용은 [표 1]과 같다.

아이폰과 사파리는 애플의 제품으로 스마트폰인 아이폰에서는 SMS를 비롯한 통화 시간 및 상대방에 대한 정보, 멀티 메일 첨부파일 내용 등 스마트폰 통신 과정에서 생성되는 정보를 SQLite를 통해 저장하고 있으며 웹브라우저인 사파리는 HTTP 웹페이지를 탐색하는 과정에서 발생하는 캐시(cache) 정보를 저장하는데 SQLite를 사용한다[6]. 모질라의 웹브라우저

〔표 1〕 SQLite를 사용하는 소프트웨어

응용프로그램	저장 데이터 유형	파일 경로 (Windows XP 기준)
Safari	Web cache	%USERPROFILE%\Local Settings\Application Data\Apple Computer\Safari\Cache.db
i-phone	스마트폰 통신 내용	%USERPROFILE%\Local Settings\Application Data\Apple Computer\MobileSync\Backup\〈랜덤값〉 폴더에 아이폰 데이터 백업시 〈랜덤값〉.mdata 파일로 저장. (다른 데이터 파일도 같은 형식으로 저장되기 때문에 시그니쳐를 검사하여 SQLite 파일 여부를 파악 해야 함)
Fire fox	Web History	%USERPROFILE%\Application Data\Mozilla\Firefox\Pro파일s\〈랜덤값〉\places.SQLite
	Web Cookie	%USERPROFILE%\Application Data\Mozilla\Firefox\Pro파일s\〈랜덤값〉\cookies.SQLite
Chrome	Web Cache	%USERPROFILE%\Local Settings\Application Data\Google\Chrome\User Data\Default\Cache\data_0
	Web History	%USERPROFILE%\Local Settings\Application Data\Google\Chrome\User Data\Default\History
	Web Cookie	%USERPROFILE%\Local Settings\Application Data\Google\Chrome\User Data\Default\Cookies
Android	스마트폰 통신 내용	/data/data/com.android.〈Software Name〉/databases
Skype	메신저 통신 내용	%USERPROFILE%\Application Data\Skype\〈스카이프 사용자 아이디〉\main.db

인 파이어폭스의 경우 모질라에서 자체적으로 개발한 파일 포맷인 모크(Mork)를 꾸준히 SQLite 파일로 대체하고 있으며 파이어폭스3 에서는 웹히스토리와 웹쿠키 정보를 SQLite를 통해 저장하고 있다. 구글의 안드로이드는 임베디드 기기를 위한 운영체제와 미들웨어 그리고 핵심 어플리케이션등을 포함하고 있는 소프트웨어로 핵심 어플리케이션에서 주로 SQLite를 이용하여 데이터를 저장하고 있다. 또한 크롬은 2008년 12월 11월 첫 안정화 버전이 나온 구글의 웹브라우저로 초기 버전에서부터 SQLite를 통하여 캐시, 히스토리, 쿠키 정보를 저장하고 있다. 마지막으로 이 베이옥션의 스카이프는 다양한 방법으로 상대방과 통신할 수 있는 기회를 제공해 주는 메신저 소프트웨어이다. 가입에 대한 제약이 없고 모든 통신은 인터넷으로 이루어지기 때문에 전 세계에 걸쳐 많은 회원을 보유하고 있다. 인터넷 전화, 문자 채팅 등의 방법으로 등록된 상대방과의 통신을 제공하며 채팅 내역, 파일 전송 정보 및 통신 상대에 대한 정보를 모두 SQLite 파일로 저장하고 있다.

3.3 응용프로그램에서의 레코드 삭제

SQLite를 사용하는 소프트웨어는 레코드를 삭제할 경우 각 소프트웨어별로 삭제된 데이터를 관리하는 정책이 다르다. 정책은 〔표 2〕와 같이 크게 세 가지로 분류된다. 첫째로 데이터를 삭제하면 0으로 덮어쓰는 경우가 있다. 웹브라우저인 파이어폭스 3, 사파리, 크롬이 이와 같은 정책을 사용하며 이 경우 삭제된 영역을 파악하여도 데이터를 복구할 수 없다. 둘째로 삭제된 영역 자체를 제거하는 경우가 있는데 아이폰에서 데이터를 소규모로 삭제할 때 발생하는 경우이며 이 경우 삭제행위가 발생했다는 흔적도 찾을 수 없다. 마지막으로 데이터 영역을 비할당 영역으로 구분하고 데이터는 그대로 남기는 경우가 있다. 스카이프 메신저에서 삭제된 문자 채팅 메시지와 아이폰에서 그룹 단위로 삭제된 문자 데이터가 이와 같은 정책에 따라 삭제된 후에도 파일에 남아있게 되고, 덮어 쓰이지 않은 경우 복구가 가능하다.

〔표 2〕 응용프로그램별 삭제데이터 관리 정책

삭제 데이터 관리 정책	복원 가능 여부	대표 응용프로그램
0으로 덮어씀	불가능	파이어폭스3, 사파리, 크롬
삭제된 영역 제거	불가능	아이폰 OS (문자 개별삭제)
비할당 영역으로 구분	가능	스카이프, 아이폰 OS (문자 그룹 삭제)

IV. SQLite 데이터베이스 파일 구조

데이터베이스 파일에서 삭제된 영역을 분류하고 해당 영역에서 레코드를 복원하기 위해선 데이터베이스 파일이 가지는 구조를 정확하게 파악하여야 한다. 본 논문에서는 SQLite 홈페이지에 공개된 SQLite 데이터베이스 파일의 구조를 토대로 레코드 삭제 및 변경 시 데이터베이스 파일에 가해지는 변화를 분석하여 복구 기법을 완성 하였다. 이 절에서는 복구 기법을 이해하기 위해 알아야 하는 데이터베이스 파일의 구조를 설명한다.

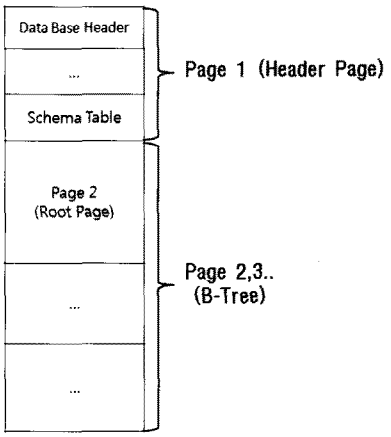
4.1 SQLite 데이터베이스 파일

SQLite 파일의 전체 구조는 [그림 1]과 같으며 파일의 시그니쳐는 [그림 2]에서와 같이 '0x53 51 4c 69 74 65 20 66 6f 72 6d 61 74 20 33 00' 이다. SQLite 파일은 특정한 확장자가 없기 때문에 분석하고자 하는 파일이 SQLite 파일인지 여부를 시그니쳐를 통해 식별해야 한다.

SQLite는 페이지라는 저장 단위를 사용하여 데이터를 저장한다. 페이지의 크기는 [그림 2]에서와 같이 파일 오프셋 16에 2바이트 빅 엔디안 형식으로 명시되어 있으며 기본 크기는 0x400이다. 파일의 최상단에 나타나는 페이지를 헤더페이지라 하는데, 헤더페이지의 상단엔 시그니쳐를 포함한 데이터베이스 파일의 헤더가 존재하며 헤더페이지의 하단엔 데이터베이스에 존재하는 테이블의 정보를 가진 스키마 테이블(Schema Table)이 존재한다.

헤더페이지 이후에 존재하는 페이지는 B-트리로 구성되어 있으며, B-트리에는 크게 인덱스 B-트리와 데이터의 내용을 저장하는 테이블 B-트리가 존재한다.

SQLite는 셀이라는 최소 저장단위를 사용하며 셀의 구조와 저장하는 데이터의 내용은 Internal 페이지와 Leaf 페이지에 따라 달라진다. Internal 페이지는 트리의 중간단계에 위치한 페이지로 이 곳에 있는 셀은 하위 페이지를 찾아가기 위한 포인터를 담고



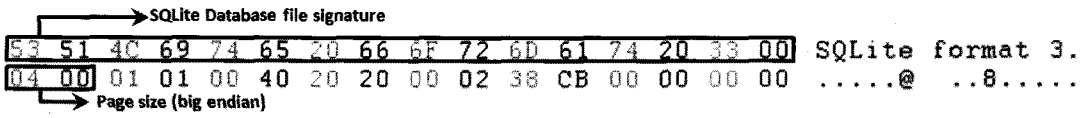
[그림 1] SQLite 파일 구조

있다. Leaf 페이지는 트리의 최 하단에 위치하여 하위 페이지를 갖지 않는 페이지로 이 곳에 있는 셀은 데이터베이스의 레코드를 담고 있다.

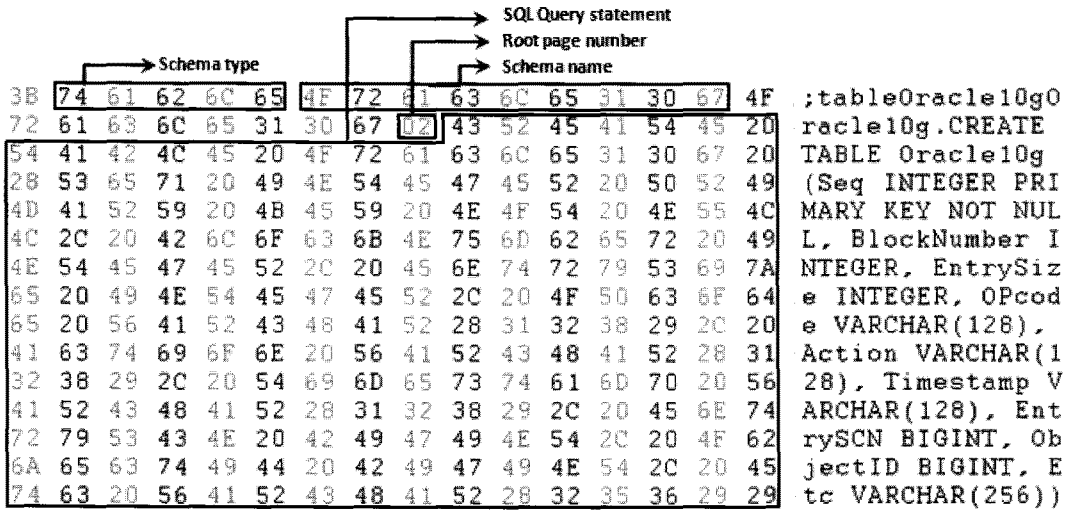
4.2 스키마 테이블

헤더 페이지에 존재하는 스키마 테이블에는 응용프로그램에서 SQLite 파일을 구성하는 테이블, 인덱스, 트리거에 관한 정보가 들어 있다. 스키마 테이블의 형식은 [그림 3]과 같으며 스키마 타입에는 스키마의 종류가 테이블, 인덱스, 트리거 중 무엇인지 명시되어 있다. 스키마 이름에는 생성된 스키마의 이름이 들어있으며 이름에 대한 복사본이 이어서 존재하고 이후로 1 바이트 정수 형태로 Root 페이지의 번호가 있다. Root 페이지 번호 이후로 나타나는 문자열은 스키마가 생성될 당시 SQLite로 전달되었던 SQL 쿼리문이다. 삭제된 레코드를 복원하기 위해서는 스키마 테이블 중 테이블 형식의 스키마에서 SQL 쿼리문을 해석하여 테이블에 관한 정보를 얻어야 한다. 스키마 테이블을 통해 얻을 수 있는 테이블의 정보는 테이블 이름, 필드의 타입과 이름, 필드의 개수이다.

SQLite의 필드 타입은 INTEGER, TEXT, BLOB, NUMERIC의 네 가지가 존재하며 VARCHAR(125), BIG INT, DATE등 표준 SQL 쿼리



[그림 2] SQLite 헤더



〔그림 3〕 SQLite 스키마 테이블

에 존재하는 데이터 타입은 SQLite의 네 가지 데이터 타입 중 각각 적당한 타입으로 변환된다. 변환되는 규칙은 SQLite 홈페이지에서 확인할 수 있다[4].

4.3 가변 길이 정수

SQLite는 정수를 저장할 때 〔표 3〕과 같이 그 크기에 따라 1~9바이트의 가변적인 길이로 저장한다. 가변 길이 정수 포맷은 〔표 3〕에서와 같이 1~8바이트의 가변 길이 정수에서는 각 바이트의 하위 7비트가 정수를 저장하는데 사용되며 9바이트의 가변 길이 정수 마지막 바이트는 8비트 모두를 수 표현에 사용하여 총 64비트로 수를 표현한다.

이처럼 SQLite에서는 정수가 크기에 따라 가변적인 길이로 저장됨에 따라 삭제된 레코드를 분석함에 있어서 각 필드를 정확하게 구분하는데 어려움이 생길 수 있다. 따라서 삭제된 레코드를 복원하기 위해서는 각 셀마다 가변적인 길이를 갖는 영역에 대한 정확한 길이를 파악해야 한다.

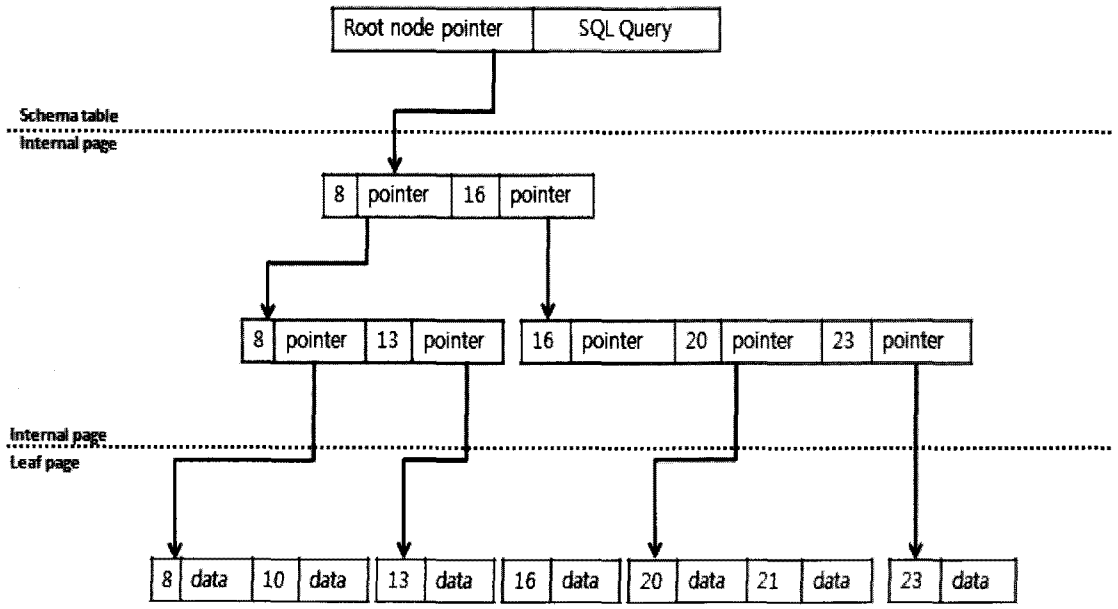
4.4 테이블 B-트리

테이블 B-트리는 SQLite 파일에 존재하는 B-트리 중 실제 데이터를 저장하기 위한 목적으로 존재하며 〔그림 4〕와 같은 구조로 되어있다.

Root 페이지의 번호는 〔그림 3〕에서와 같이 스키

〔표 3〕 Variable length integer format

Byte	Value range	Bit pattern
1	7 bit	0XXXXXXX
2	14 bit	1XXXXXXX 0XXXXXXX
3	21 bit	1XXXXXXX 1XXXXXXX 0XXXXXXX
4	28 bit	1XXXXXXX 1XXXXXXX 1XXXXXXX 0XXXXXXX
5	35 bit	1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 0XXXXXXX
6	42 bit	1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 0XXXXXXX
7	49 bit	1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 0XXXXXXX
8	56 bit	1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 0XXXXXXX
9	64 bit	1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX 1XXXXXXX XXXXXXXX

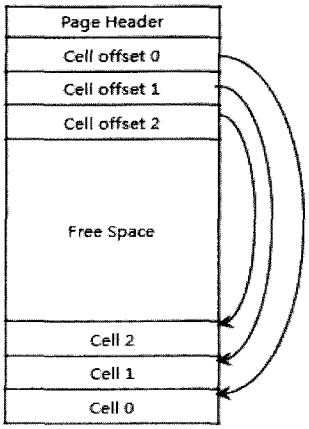


〔그림 4〕 SQLite B-트리 구조

마 테이블의 SQL 쿼리문 바로 앞에 1바이트 정수로 명시되어 있으며 Root 페이지는 경우에 따라 Internal 페이지와 Leaf 페이지가 되기도 한다. Internal 페이지는 다음 하위 페이지를 찾아가기 위한 정보를 담고 있으며, 실제 데이터는 Leaf 페이지에 저장된다. 본 논문에서는 실제 데이터에 접근하여 삭제된 영역을 파악, 복원하는 방법을 설명하기 위해 Leaf 페이지의 내용에 집중하여 아래 내용을 기술한다.

4.5 페이지 구조와 비할당 영역

SQLite 파일에 존재하는 모든 페이지는 Internal 페이지와 Leaf 페이지 모두 〔그림 5〕와 같은 구조로 구성되어 있다. 페이지 헤더에 이어 셀의 위치를 파악할 수 있는 오프셋 값이 2바이트 빅엔디안 정수로 나열되어 있으며 명시된 오프셋의 위치에 실제 데이터를 담은 셀이 위치한다. 셀은 페이지의 최 하단부터 차례로 사용된다. 셀 오프셋과 셀 사이의 영역은 비할당 공간(Free space)이며, 초기 생성시 0으로 채워진다. Leaf 페이지 내부엔 비할당 공간 외에 비할당 블록(Free Block)이라는 비할당 영역이 존재할 수 있는데 이는 삭제된 셀이 삭제되기 이전에 위치하던 영역으로 차후 새로운 셀이 할당될 때 이용하기 위한 용도로 관리된다.

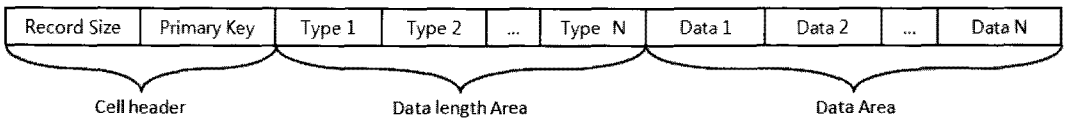


〔그림 5〕 SQLite 페이지 구조

4.5.1 페이지 헤더

4.5.1.1 Leaf 페이지 헤더

〔표 4〕의 내용과 같이 SQLite 데이터베이스 파일의 Leaf 페이지의 헤더는 12바이트로 구성되며 첫 바이트는 페이지 플래그로 0x05의 값을 가진다. 다음 오프셋 1~2는 2바이트 빅엔디안 정수로 첫 번째 비할당 블록의 오프셋을 나타낸다. 다음으로 오프셋 3~4는 페이지에 존재하는 셀의 개수를 2바이트 빅엔



〔그림 6〕 Leaf 셀 구조

디안 정수로 나타낸다. 셀의 개수가 0일 경우 이 페이지를 비할당 페이지라 한다. 오프셋 5~6은 첫 번째로 나타나는 셀의 오프셋을 나타낸다. 다음으로 오프셋 7은 비할당 블록 중 3바이트 이하의 크기를 가진 비할당 블록의 개수를 1바이트 정수 형태로 나타내고 있다.

4.5.2 셀

SQLite에는 셀이라는 최소 저장 단위가 존재하며 테이블 B-트리의 Leaf 페이지에 존재하는 셀에는 데이터베이스의 레코드가 포함된다. 삭제된 레코드를 복원하기 위해서는 Leaf 페이지의 셀 구조와 셀이 가지는 규칙을 파악해야 한다.

4.5.2.1 Leaf 셀

Leaf 페이지에 존재하는 Leaf 셀은 담고 있는 레코드를 표현하기 위해 〔그림 6〕과 같이 셀 헤더(Cell header), 데이터 길이 영역(Data length area), 데이터 영역(Data area)의 세 가지 영역으로 레코드를 관리한다. 셀 헤더에 있는 가변 길이 정수 형태의 기본키는 각 레코드마다 고유한 값을 가지며, 데이터 길이 영역에는 데이터 영역에 위치하는 레코드의 각 필드에 대한 바이트 길이 정보가 〔표 5〕에 따라 담겨 있다.

데이터 길이 영역의 크기는 (필드의 개수) + α 가 되는데, 여기서 α는 가변적인 값으로 TEXT와 BLOB 타입은 데이터의 길이가 56 바이트를 넘어갈 경우 데이터 길이 영역에 2바이트로 길이를 표현하기에 발생하는 값이다. 데이터 길이 영역에 명시된 길이에 따라 각 필드의 데이터가 데이터 영역에 쓰여 있다. 이와 같이 셀 헤더의 기본키와 데이터 길이 영역의 바이트길이가 셀마다 가변적이기 때문에, 레코드가 존재하는 데이터 영역의 내용을 정확하게 파악하기 위해선 각 영역의 정확한 길이를 파악하는 것이 중요하다. 데이터 길이의 파악은 추측과 검증을 반복하는 과정에 따라 이루어질 수 있으며 구체적인 내용은 4.3에 기술하였다.

〔표 4〕 Internal 페이지 헤더 내용

Offset	내용
0	Page flag : 0x0D
1-2	첫 번째 비할당 블록 오프셋
3-4	페이지 내의 셀 개수
5-6	첫 번째 나타나는 셀 오프셋
7	3 바이트 이후 비할당 블록 개수

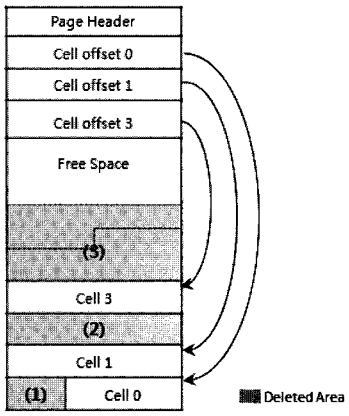
V. SQLite의 삭제된 레코드 복구 기법

5.1 삭제된 영역의 레코드 복원 과정

삭제된 레코드를 복구하기 위해서는 먼저 스키마 테이블을 통해 테이블의 필드의 개수와 각 필트의 타입을 파악해야 한다. 필드의 타입과 개수는 레코드 복원 과정에서 Leaf셀에 존재하는 길이가 가변적인 영역을 구분하는데 이용된다. 이후 페이지를 순차적을 탐색하며 페이지 내의 모든 비할당 영역(비할당 공간 + 비할당 블록)을 파악하여 데이터를 추출한다. 페이지 내의 모든 비할당 영역을 검사한 후에는 다음 페이지를 탐색하며, 모든 페이지를 탐색하면 복원과정이 종료된다. 삭제된 영역의 관점에서는 비할당 영역이라고 구분됐지만 값이 0이 아닌 영역은 모두 삭제된 영

〔표 5〕 Type 영역의 표시 형식

Value	Data Type	Data Size
0	NULL	0
N (N=1-4)	Signed Integer	N
5	Signed Integer	6
6	Signed Integer	8
7	IEEE float	8
8-11	Reserved	
N>12 (N:even)	BLOB	(N-12)/2
N>13 (N:odd)	TEXT	(N-13)/2



〔그림 7〕 페이지 내 삭제된 영역

역이라 간주할 수 있다. 삭제된 영역에는 하나 혹은 다수의 레코드가 존재하는데, 레코드가 손상되지 않았다면 모두 복구 가능하다.

SQLite 파일에서 삭제된 영역은 〔그림 7〕처럼 세 가지 경우로 구분되며 각 영역의 특징은 아래와 같다.

(1) 삭제된 후 다른 데이터가 덮어 쓰여 데이터의 일부가 남아있는 경우
(2) 삭제된 영역에 하나의 레코드가 독립적인 상태로 존재하는 경우
(3) 삭제된 영역에 레코드가 2개 이상 존재하는 경우

각 영역은 특징에 따라 복원 가능 여부가 다르고 복원 절차도 다르다. 먼저 (1)번과 같은 경우 〔그림 8〕의 복원과정에서 Inspect Free Block의 Case 1에 해당하며 별다른 절차 없이 바로 복원이 가능하다. 반면 (2)번과 같은 경우 〔그림 8〕의 Case 2에 해당하며 (1)번과 같은 형태로 데이터를 분할한 후 복원이 가능하다. 반면 (3)번과 같은 경우 셀 내의 각 영역에 관한 길이를 추측할 수 없어 복원이 불가능하다.

5.2 데이터베이스 파일 내 삭제된 영역 파악

삭제된 영역에 접근하기 위해선 먼저 Leaf 페이지를 탐색해야 한다. Leaf 페이지는 페이지의 첫 바이트에 0x0D의 값을 쓰고 있으므로 파일을 페이지 크기 단위로 탐색하면서 첫 바이트를 식별하여 다른 페이지와 구분할 수 있다.

4.5.1.1에 기술된 페이지 헤더 정보에 따르면 Leaf 페이지는 바이트 오프셋 2~3에 2바이트로 첫 번째 비할당 블록의 오프셋을 저장한다. 한 개의 페이지 내에는 0개 이상의 비할당 블록이 존재하며, 각각의 블록은 자신의 바로 다음에 나타나는 비할당 블록의 오프셋을 상위 2바이트에 저장하면서 비할당 블록 체인을 구성하고 있다. 마지막 비할당 블록은 상위 2바이트에 0x0000값을 가지며 자신이 비할당 블록체인의 마지막임을 나타낸다. 이렇듯 SQLite는 비할당 블록체인을 이용하여 페이지 내 모든 비할당 블록을 탐색할 수 있도록 설계되어 있다.

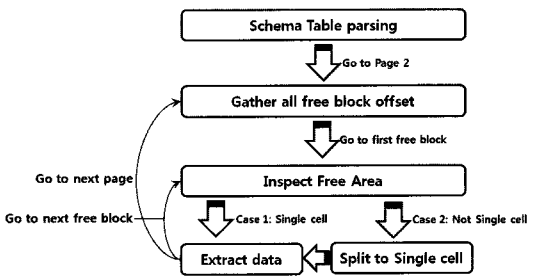
삭제된 셀은 보통 비할당 블록에 존재하며, 경우에 따라서는 정상 셀 보다 앞에 존재하여 비할당 공성 시 0으로 되어있기 때문에, 비할당 공간에 0이 아닌 영역을 검색함으로써 삭제된 영역을 파악할 수 있다.

5.3 가변 길이 영역의 길이 파악과 레코드 복원

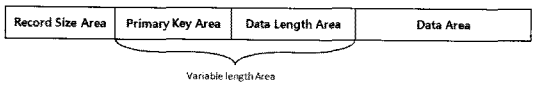
〔그림 8〕에서 삭제된 영역이 Case 1로 분류되면, 삭제된 영역 내에 한 개의 레코드만이 독립적으로 존재한다고 판단되는 것이기 때문에, 곧바로 각 필드를 구분하여 복원과정을 진행한다.

SQLite는 레코드를 삭제할 경우 해당 레코드를 담고 있던 셀의 영역을 비할당 블록으로 구분하기 위해 셀의 첫 4바이트의 정보 중 상위 2 바이트는 다음 비할당 블록의 오프셋으로, 하위 2바이트는 현재 비할당 블록의 길이로 갱신한다. 이와 같은 이유로 삭제된 셀의 첫 4바이트는 정상적으로 존재할 때와 다른 값을 가지고 있으며 정상적인 데이터와는 다른 방법으로 셀의 각 필드를 구분해야 한다.

Leaf 셀의 헤더에서 기본키 영역엔 기본키 값이 가변 길이 정수 형태로 존재하는데 〔그림 10〕과 같이 셀이 삭제되면서 갱신되는 정보에 의해 일부가 덮어 쓰여지기 때문에 삭제된 셀에서 기본키 영역의 정확한 길이를 알 수 없다. 또한 데이터 길이 정보 영역에선 TEXT 타입과 BLOB 타입의 경우 1 또는 2바이트로



〔그림 8〕 삭제데이터 복원 과정

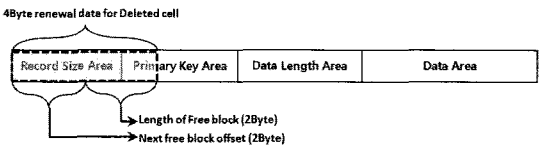


〔그림 9〕 셀 내의 가변길이 영역

해당 필드에 있는 데이터의 길이를 표현하기 때문에 데이터 Length 영역도 그 길이가 삭제된 셀마다 가변이다.

이 때문에 삭제된 셀을 복원하기 위해선 〔그림 9〕에서 표현된 가변 길이 영역인 기본키 영역과 데이터 길이 영역의 정확한 길이를 알아야 한다. 길이는 추측과 검증과정을 반복하여 파악할 수 있다. 검증을 위해선 먼저 셀의 길이에 대한 정보를 입력받아야 하며 기본키 영역의 추측 길이를 1바이트 로 시작하여 차례로 늘려가면서 검증을 진행한다.

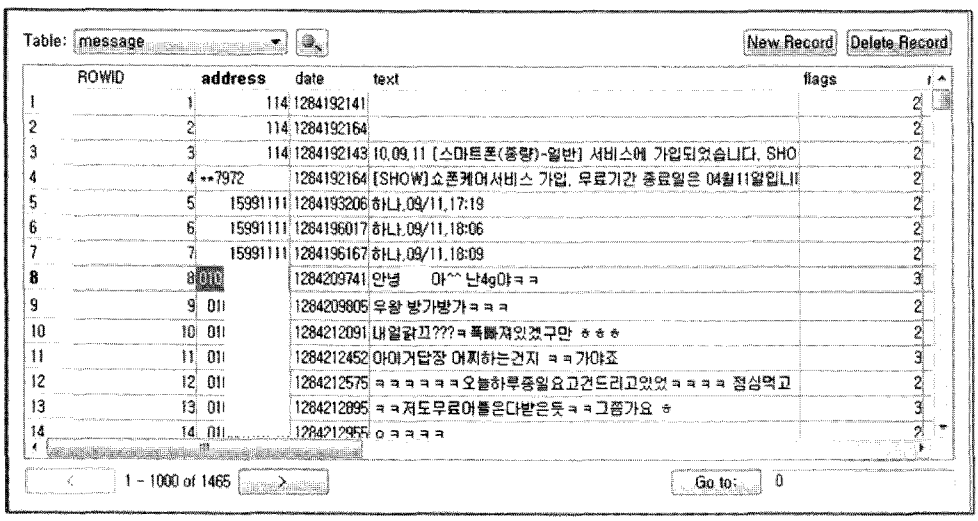
검증 과정은 추측된 기본키 길이를 토대로 셀의 길이를 다시 계산하여 처음 입력받은 셀의 길이와 비교하는 과정으로 이루어진다. 기본키 영역의 길이가 추측되면 이를 토대로 기본키 영역 바로 다음에 위치하는 데이터 길이 영역의 길이를 (필드의 개수) + α 로 계산할 수 있는데 α는 56바이트 이상의 길이를 가지는 TEXT와 BLOB형태의 필드 개수이다. 이후 계산된 값을 처음 넘겨받은 셀의 길이와 비교하여 두 값이 같을 경우 추측한 기본키 영역의 길이와 데이터 길이 영역의 길이를 옳은 값이라 판단하여 복원을 진행한다. 비교한 값이 서로 다를 경우 기본키 영역 길이의 추측 값을 1 늘려서 위 과정을 반복하고, 기본키 영역의 추측 길이가 가변 길이 정수의 최대 길이인 9바이트를 넘었을 경우 현재 셀은 일부가 덮어 쓰여 손상되었거나 셀의 전체 길이가 올바르게 입력되지 않았다고 판단하여 복원을 중단한다. 복원이 중단되면 해당 삭제된 영역이 손상되었거나 다수의 레코드를 포함하고 있는 것으로 판단한다.



〔그림 10〕 삭제된 셀의 상위 4바이트 변경 정보

5.4 복원이 이루어지지 않은 경우

5.3에 기술한 방법에 따라 복원을 진행하였을 때 복원이 이루어지지 않는다면 이는 〔그림 8〕에서 Case 2에 해당하는 영역으로 손상되었거나 다수의 레코드를 포함하는 경우이다. 두 가지 경우에는 셀의 길이를 알 수 있는 정보가 주어지지 않기 때문에 셀의 길이를 최초 n (n : 필드의 개수)바이트로 추측하여 차례로 1씩 늘려가며 5.3의 방법에 따라 복원을 다시 시도한다. 해당 영역이 손상되지 않고 레코드를 포함하고 있다면 셀의 추측길이를 늘려가던 중 복원이 이루어지게 되며 복원된 결과를 추출한다. 이후 복원된 영역을 제외시키고 남아있는 영역이 없을 때까지 나머지 데이터에 대해 같은 과정을 반복한다. 하지만 셀의 추측길이를 삭제된 영역의 크기만큼 늘렸을 경우에



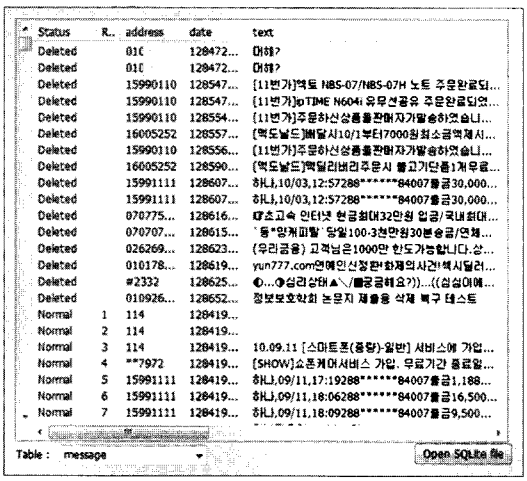
〔그림 11〕 SQLite Database Browser로 확인한 데이터베이스 파일의 내용

도 복원된 레코드가 없다면 해당 영역이 손상되었다고 판단한다.

VI. 실험 결과

5장에 기술된 복원 기법을 토대로 삭제된 레코드를 복구하는 도구를 구현하였으며, 지정한 폴더 내 SQLite 파일을 파악하여 파일에 존재하는 삭제된 레코드를 복원한 내용을 확인하였다. 복원에 사용된 SQLite 데이터베이스 파일은 실제로 사용 중인 아이폰 4에서 추출한 문자 데이터를 저장하고 있는 SQLite 데이터베이스 파일이다. 아이폰은 3장의 〔표 1〕과 〔표 2〕에서 정리한 바와 같이 데이터 백업 시 %USERPROFILE%\Local Settings\Application Data\Apple Computer\MobileSync\Backup\〈랜덤값〉 폴더에 백업 데이터를 저장하며 문자 메세지를 그룹단위로 삭제한 경우 일부 데이터를 데이터베이스 파일에 남긴다. 본 실험에서는 광고나 통보메세지등을 일괄적으로 삭제한 후 데이터 복원을 시도하였다.

〔그림 11〕 은 SQLite Database Browser를 이용하여 아이폰에 저장된 문자 데이터를 확인한 내용이다. 〔그림 12〕는 SQLite 레코드 복구 도구를 사용하여 삭제된 아이폰의 문자 데이터를 복구한 내용이다. 광고, 통보 메시지 이외에도 "정보보호학회 논문지 제출용 삭제 복구 테스트"라는 내용의 메시지를 임의로 전송한 후 삭제 테스트를 한 내용도 〔그림 12〕를 통해 확인할 수 있다. Status 열에 "Deleted"라는 값을 가진 행이 삭제된 후 복원된 데이터를 나타낸 것



〔그림 12〕 구현된 도구로 확인한 데이터베이스 파일의 내용

이며, "Normal"이라는 값을 가진 행은 정상적으로 남아있는 레코드임을 표시한 것이다. "Normal"이라는 값을 가진 행은 실제로 〔그림 11〕의 일부분과 내용이 일치하는 것을 확인할 수 있다. 삭제된 후 복원된 데이터를 보면 Status열의 바로 옆에 기본 키가 출력되지 않은 것을 확인할 수 있는데, 이는 5.3에서 설명한 바와 같이 삭제된 영역을 관리하는 4바이트 갱신정보에 의해 기본 키 영역의 일부가 덮어씌워졌기 때문이다. 기본 키를 제외한 나머지 필드에 대해서는 모두 복원이 가능하며 본 실험에서 그 결과를 확인하였다.

VII. 결 론

SQLite의 사용이 많아짐에 따라 SQLite의 포렌식 관점의 중요성이 매우 커지고 있다. 특히 삭제된 영역의 복구는 현재까지 연구가 매우 미흡하다.

본 논문에서는 SQLite 파일의 포맷의 분석을 통하여 파악된 비할당 영역에 잔존하는 데이터를 효과적으로 복구하는 기법을 제안하였다. 본 논문에서 제안하는 복구기법은 SQLite를 사용하는 모든 응용프로그램에 적용 가능하다. 실험을 통해 삭제된 레코드가 잔존하는 경우 레코드가 복원되는 것을 확인하였으며 제안된 방법은 휴대용 디지털 기기의 사용이 증가함에 따라 SQLite의 사용이 늘고 있는 시점에서 디지털 포렌식 조사를 진행함에 있어 SQLite 파일에서 의미있는 정보를 추출하는데 많은 도움이 될 것이다.

참고문헌

〔1〕 Murilo Tito Pereira, "Forensic analysis of the Firefox 3 Internet history and recovery of deleted SQLite records," Digital Investigation, pp. 93-103 , March. 2004.

〔2〕 Chris Newman, SQLite, Macmillan ComputerPub, October. 2004

〔3〕 SQLite, "SQL As Understood By SQLite," http://www.sqlite.org/lang.html

〔4〕 SQLite, "Datatypes In SQLite Version 3," http://www.sqlite.org/datatype3.html

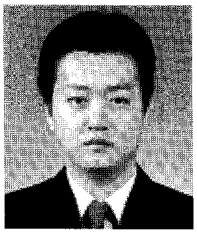
〔5〕 SQLite, "Well-Known Users of SQLite," http://www.sqlite.org/famous.html

〔6〕 SQLite, "iPhone OS Data Management ," http://developer.apple.com/technologie

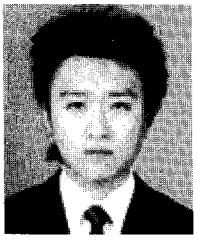
s/iphone/data-management.html
[7] Theo Haerder, "Principles of transaction-oriented database recovery," ACM Computing Surveys (CSUR), pp. 287-317, December. 1983
[8] Joost S.M. Verhofstad, "Recovery Techniques for Database Systems," ACM Computing Surveys (CSUR), pp. 167-195 , June. 1978
[9] R.A Crus, "Data recovery in IBM database 2," IBM Systems Journal, Vol(23), pp. 178-188 . March. 1984
[10] Mike Owens,Michael Owens, The definitive guide to SQLite, Apress , May. 2006

〈著者紹介〉

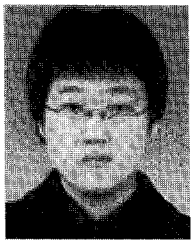
전 상 준 (Gil-dong Hong) 학생회원
2010년 2월: 고려대학교 정보경영공학과 공학사
2010년 2월 ~ 현재: 고려대학교 디지털 포렌식 연구센터 석박사통합과정
〈관심분야〉 디지털 포렌식



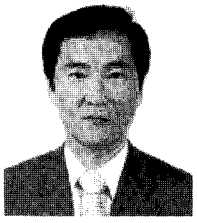
변 근 덕 (KeunDuck Byun) 학생회원
2004년 2월: 아주대학교 정보및컴퓨터공학부 졸업
2006년 2월: 고려대학교 정보경영공학전문대학원 석사
2006년 ~ 현재: 고려대학교 정보경영공학전문대학원 박사과정
〈관심분야〉 임베디드 포렌식, 역공학, 윈도우 포렌식, 이미지 포렌식



방 제 완 (Jewan Bang) 정회원
2007년 2월: 한세대학교 정보통신공학과 졸업
2007년 3월 ~ 현재: 고려대학교 정보경영공학전문대학원 석박사통합과정
〈관심분야〉 디지털 포렌식, 역공학 분석, 임베디드 시스템



이 근 기 (Keun-Gi Lee) 학생회원
2007년 2월: 부경대학교 전자컴퓨터정보통신공학부 졸업
2010년 8월: 고려대학교 정보경영공학전문대학원 석사
2010년 - 현재: 고려대학교 정보경영공학전문대학원 박사과정
〈관심분야〉 라이브 포렌식, 모바일 포렌식, 기업 포렌식



이 상 진 (SangJin Lee) 종신회원
1987년: 고려대학교 학사
1989년: 고려대학교 석사
1994년: 고려대학교 박사
1989년 ~ 1999년: ETRI 연구원 역임
1992년: 국가안전기획부장 표창
1999년 ~ 현재: 고려대학교 정교수
〈관심분야〉 디지털 포렌식, 모바일 포렌식, 심층 암호, 해쉬 함수