(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년09월19일
(11) 등록번호 10-1657592
(24) 등록일자 2016년09월08일

(51) 국제특허분류(Int. Cl.)
G06F 11/14 (2006.01) *G06F 17/30* (2006.01)
(52) CPC특허분류
G06F 11/1435 (2013.01)
G06F 11/1402 (2013.01)
(21) 출원번호 10-2015-0121743
(22) 출원일자 2015년08월28일
심사청구일자 2015년08월28일
(56) 선행기술조사문헌
'MySQL 데이터베이스의 삭제된 레코드 복구 기법', 남국재웅 외 4명, 디지털포렌식 연구 제7권제1호(2013.12.)*
'데이터베이스 내 삭제된 레코드의 복원 방법에 관한 연구', 박수영, 고려대학교 정보보호대학원 석사학위논문(2013.12.)*
KR101547466 B1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
대한민국(관리부서 대검찰청)
서울특별시 서초구 반포대로 157 (서초동)
(72) 발명자
이상진
서울특별시 종로구 사직로8길 20, 101동 1503호 (내수동, 파크팰리스)
노우선
대전광역시 중구 서문로 96, 204동 1102호 (문화동, 센트럴파크2단지아파트)
(뒷면에 계속)
(74) 대리인
특허법인충현

전체 청구항 수 : 총 6 항 심사관 : 이동하

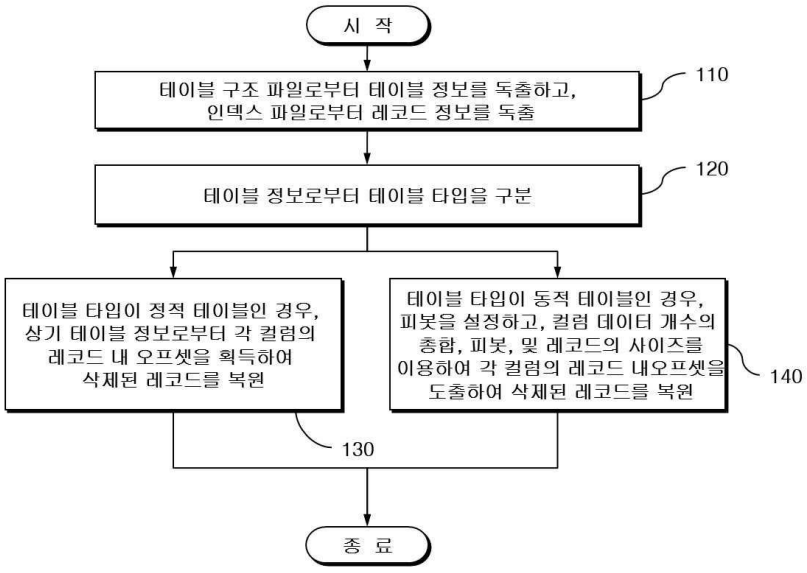
(54) 발명의 명칭 **삭제된 MySQL MyISAM database의 데이터에 대한 범용 복원기법**

(57) 요 약

본 발명은 레코드를 복원하는 방법에 관한 것으로서, 레코드를 복원하는 방법은, 테이블 구조 파일로부터 테이블 정보를 독출하고, 인덱스 파일로부터 레코드 정보를 독출하는 단계와, 상기 테이블 정보로부터 테이블 타입을 구분하는 단계와, 상기 테이블 타입이 정적 테이블인 경우, 상기 테이블 정보로부터 각 컬럼의 레코드 내 오프셋을

(뒷면에 계속)

대 표 도 - 도1



획득하여 삭제된 레코드를 복원하는 단계와, 상기 테이블 타입이 동적 테이블인 경우, 20번째 바이트를 피봇으로 설정하는 단계와, 컬럼 데이터 개수 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은지 판단하는 단계 및 상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은 경우, 현재 피봇을 첫 번째 컬럼의 레코드 내 시작 오프셋으로 도출하고 삭제된 레코드를 복원하는 단계를 포함하고, 상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 다른 경우, 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같아질 때까지 피봇을 1씩 증가시키면서 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복하는 단계를 포함하고, 상기 피봇은 첫 번째 컬럼의 레코드 내 시작 오프셋으로 가정되는 오프셋인 것을 특징으로 함으로써, 트랜잭션 로그 파일에 비종속적으로 데이터베이스 파일에서 테이블 스키마와 삭제된 레코드를 추출할 수 있다.

(52) CPC특허분류
G06F 17/30289 (2013.01)
G06F 17/30312 (2013.01)

(72) 발명자
정두원
서울특별시 노원구 섬밭로 123, 304동 903호 (공릉동, 라이프아파트)
최재문
서울특별시 금천구 독산로86길 23, 403호 (독산동, 대광파크)
강철훈
서울특별시 서초구 효령로2길 30, 104호 (방배동, 방배엔스위트)
이경민
경기도 화성시 동탄지성로 17, A동 2301호 (반송동, 풍성위버폴리스)

이 발명을 지원한 국가연구개발사업
과제고유번호 2012M3A2A1051106
부처명 미래창조과학부
연구관리전문기관 한국연구재단
연구사업명 원천기술개발사업
연구과제명 대용량 저장장치 조사용 포렌식 시스템 개발
기 여 율 1/1
주관기관 고려대학교 산학협력단
연구기간 2015.08.01 ~ 2016.07.31

명 세 서

청구범위

청구항 1

레코드를 복원하는 방법에 있어서,

테이블 구조 파일로부터 테이블 정보를 독출하고, 인덱스 파일로부터 레코드 정보를 독출하는 단계;

상기 테이블 정보로부터 테이블 타입을 구분하는 단계;

상기 테이블 타입이 정적 테이블인 경우, 상기 테이블 정보로부터 각 컬럼의 레코드 내 오프셋을 획득하여 삭제된 레코드를 복원하는 단계;

상기 테이블 타입이 동적 테이블인 경우, 20번째 바이트를 피봇으로 설정하는 단계;

컬럼 데이터 개수 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은지 판단하는 단계; 및

상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은 경우, 현재 피봇을 첫 번째 컬럼의 레코드 내 시작 오프셋으로 도출하고 삭제된 레코드를 복원하는 단계를 포함하고,

상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 다른 경우, 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같아질 때까지 피봇을 1씩 증가시키면서 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복하는 단계를 포함하고,

상기 피봇은,

첫 번째 컬럼의 레코드 내 시작 오프셋으로 가정되는 오프셋인 것을 특징으로 하는 방법.

청구항 2

삭제

청구항 3

제 1 항에 있어서,

상기 테이블 타입이 동적 테이블인 경우, 각 컬럼의 레코드 내 오프셋을 도출하여 레코드를 복원하는 단계는,

상기 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같은 경우, 복원할 수 있는 컬럼의 수가 0이 될 때까지 복원할 수 있는 컬럼의 수를 한 개씩 줄이면서 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복하는 것을 특징으로 하는 방법.

청구항 4

제 3 항에 있어서,

상기 테이블 타입이 동적 테이블인 경우, 각 컬럼의 레코드 내 오프셋을 도출하여 레코드를 복원하는 단계는,

상기 복원할 수 있는 컬럼의 수가 0인 경우, 상기 테이블 정보로부터 스킵이 가능한 컬럼이 있는지 확인하고, 스킵이 가능한 컬럼이 있는 경우, 스킵이 가능한 컬럼을 고려하여 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복하는 것을 특징으로 하는 방법.

청구항 5

제 1 항에 있어서,

상기 테이블 타입이 정적 테이블인 경우, 상기 테이블 정보로부터 각 컬럼의 레코드 내 오프셋을 획득하여 삭제된 레코드를 복원하는 단계는,

테이블 정보로부터 첫 7개의 바이트를 제외한 각 컬럼의 레코드 내 오프셋을 획득하여 삭제된 레코드를 복원하

는 것을 특징으로 하는 방법.

청구항 6

제 1 항에 있어서,

상기 레코드 정보에 포함된 삭제된 레코드의 수 및 마지막으로 삭제된 레코드의 오프셋 정보를 이용하여 삭제된 모든 레코드에 대해 복원을 수행하는 것을 특징으로 하는 방법.

청구항 7

제 1 항 및 제3항 내지 제 6 항 중 어느 한 항의 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 레코드를 복원하는 방법에 관한 것으로서, 더욱 상세하게는 트랜잭션 로그 파일에 비종속적으로 데이터베이스 파일에서 테이블 스키마와 삭제된 레코드를 추출할 수 있는 레코드 복원 방법에 대한 것이다.

배 경 기 술

[0002] 체계화된 정보 저장 수단인 데이터베이스는 전산데이터를 보관할 필요가 있는 모든 분야에서 광범위하게 사용되고 있다. 특히 기업의 경우 조직의 업무 결과물을 효율적으로 관리하기 위해 데이터베이스를 사용하고 있다. 따라서, 기업을 대상으로 하는 디지털 포렌식 수사과정에서 데이터베이스는 중요한 수집 대상이 된다.

[0003] 데이터베이스를 대상으로 증거 수집 시 정상적인 데이터를 수집하는 것도 중요하지만, 사용자가 의도적으로 민감한 데이터를 삭제할 가능성이 존재하기 때문에 삭제된 데이터를 복구하는 기법에 대해 연구가 필요하다.

[0004] 특히, 현재 MySQL MyISAM 데이터베이스 파일에서는 데이터를 복원하는 방안이 없는 문제가 있다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) 공개특허공보 제 10-2010-0134355호 (2010.12.23)

발명의 내용

해결하려는 과제

[0006] 본 발명이 해결하고자 하는 과제는 트랜잭션 로그 파일에 비종속적으로 데이터베이스 파일에서 테이블 스키마와 삭제된 레코드를 추출할 수 있는 레코드 복원 방법을 제공하는 것이다.

과제의 해결 수단

[0007] 본 발명은 상기 과제를 달성하기 위하여, 레코드를 복원하는 방법은, 테이블 구조 파일로부터 테이블 정보를 독출하고, 인덱스 파일로부터 레코드 정보를 독출하는 단계와, 상기 테이블 정보로부터 테이블 타입을 구분하는 단계와, 상기 테이블 타입이 정적 테이블인 경우, 상기 테이블 정보로부터 각 컬럼의 레코드 내 오프셋을 획득하여 삭제된 레코드를 복원하는 단계와, 상기 테이블 타입이 동적 테이블인 경우, 20번째 바이트를 피봇으로 설정하는 단계와, 컬럼 데이터 개수 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은지 판단하는 단계 및 상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은 경우, 현재 피봇을 첫 번째 컬럼의 레코드 내 시작 오프셋으로 도출하고 삭제된 레코드를 복원하는 단계를 포함하고, 상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 다른 경우, 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같아질 때까지 피봇을 1씩 증가시키면서 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복하는 단계를 포함하

고, 상기 피봇은 첫 번째 컬럼의 레코드 내 시작 오프셋으로 가정되는 오프셋인 것을 특징으로 하는 방법을 제공한다.

[0008] 본 발명의 실시예에 의하면, 상기 테이블 타입이 동적 테이블인 경우, 각 컬럼의 레코드 내 오프셋을 도출하여 레코드를 복원하는 단계는, 20번째 바이트를 피봇으로 설정하는 단계; 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은지 판단하는 단계; 및 상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은 경우, 현재 피봇을 첫 번째 컬럼의 레코드 내 시작 오프셋으로 도출하고 삭제된 레코드를 복원하는 단계를 포함하고, 상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 다른 경우, 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같아질 때까지 피봇을 1씩 증가시키면서 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복하는 것을 특징으로 하는 방법일 수 있다.

[0009] 본 발명의 실시예에 의하면, 상기 테이블 타입이 동적 테이블인 경우, 각 컬럼의 레코드 내 오프셋을 도출하여 레코드를 복원하는 단계는, 상기 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같은 경우, 복원할 수 있는 컬럼의 수가 0이 될 때까지 복원할 수 있는 컬럼의 수를 한 개씩 줄이면서 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복하는 것을 특징으로 하는 방법일 수 있다.

[0010] 본 발명의 실시예에 의하면, 상기 테이블 타입이 동적 테이블인 경우, 각 컬럼의 레코드 내 오프셋을 도출하여 레코드를 복원하는 단계는, 상기 복원할 수 있는 컬럼의 수가 0인 경우, 상기 테이블 정보로부터 스킵이 가능한 컬럼이 있는지 확인하고, 스킵이 가능한 컬럼이 있는 경우, 스킵이 가능한 컬럼을 고려하여 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복하는 것을 특징으로 하는 방법일 수 있다.

[0011] 본 발명의 실시예에 의하면, 상기 테이블 타입이 정적 테이블인 경우, 상기 테이블 정보로부터 각 컬럼의 레코드 내 오프셋을 획득하여 삭제된 레코드를 복원하는 단계는, 테이블 정보로부터 첫 7개의 바이트를 제외한 각 컬럼의 레코드 내 오프셋을 획득하여 삭제된 레코드를 복원하는 것을 특징으로 하는 방법일 수 있다.

[0012] 본 발명의 실시예에 의하면, 상기 레코드 정보에 포함된 삭제된 레코드의 수 및 마지막으로 삭제된 레코드의 오프셋 정보를 이용하여 삭제된 모든 레코드에 대해 복원을 수행하는 것을 특징으로 하는 방법일 수 있다.

발명의 효과

[0013] 본 발명에 따르면, 디지털 포렌식 과점에서 용의자가 고의적으로 데이터를 삭제하였을 경우, 해당 데이터를 복구하여 증거 및 삭제 행위를 입증하고, 회사 정책이나 데이터베이스 설정으로 인해 트랜잭션 로그를 일정 기간만 보관하거나 없는 경우 데이터 복구가 가능하다.

도면의 간단한 설명

[0014] 도 1은 본 발명의 일 실시예에 따른 레코드 복원방법의 흐름도이다.

도 2 내지 5는 본 발명의 실시예에 따른 레코드 복원방법의 흐름도이다.

도 6 내지 도 10은 본 발명의 실시예에 따른 레코드 복원방법이 적용되는 데이터베이스의 정보들의 구조를 나타낸 것이다.

발명을 실시하기 위한 구체적인 내용

[0015] 본 발명에 관한 구체적인 내용의 설명에 앞서 이해의 편의를 위해 본 발명이 해결하고자 하는 과제의 해결 방안의 개요 혹은 기술적 사상의 핵심을 우선 제시한다.

[0016] 본 발명의 일 실시예에 따른 레코드를 복원하는 방법은, 테이블 구조 파일로부터 테이블 정보를 독출하고, 인덱스 파일로부터 레코드 정보를 독출하는 단계, 상기 테이블 정보로부터 테이블 타입을 구분하는 단계, 상기 테이블 타입이 정적 테이블인 경우, 상기 테이블 정보로부터 각 컬럼의 레코드 내 오프셋을 획득하여 삭제된 레코드를 복원하는 단계, 및 상기 테이블 타입이 동적 테이블인 경우, 피봇을 설정하고, 컬럼 데이터 개수의 총합, 피봇, 및 레코드의 사이즈를 이용하여 각 컬럼의 레코드 내 오프셋을 도출하여 삭제된 레코드를 복원하는 단계를 포함하고, 상기 피봇은, 첫 번째 컬럼의 레코드 내 시작 오프셋으로 가정되는 오프셋인 것을 특징으로 한다.

[0017] 이하 첨부된 도면을 참조하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 본 발명을 용이하게 실시할 수 있는 실시 예를 상세히 설명한다. 그러나 이들 실시예는 본 발명을 보다 구체적으로 설명하기 위한 것으로, 본 발명의 범위가 이에 의하여 제한되지 않는다는 것은 당업계의 통상의 지식을 가진 자에게 자명할 것이

다.

[0018] 본 발명이 해결하고자 하는 과제의 해결 방안을 명확하게 하기 위한 발명의 구성을 본 발명의 바람직한 실시예에 근거하여 첨부 도면을 참조하여 상세히 설명하되, 도면의 구성요소들에 참조번호를 부여함에 있어서 동일 구성요소에 대해서는 비록 다른 도면상에 있더라도 동일 참조번호를 부여하였으며 당해 도면에 대한 설명시 필요한 경우 다른 도면의 구성요소를 인용할 수 있음을 미리 밝혀둔다. 아울러 본 발명의 바람직한 실시 예에 대한 동작 원리를 상세하게 설명함에 있어 본 발명과 관련된 공지 기능 혹은 구성에 대한 구체적인 설명 그리고 그 이외의 제반 사항이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우, 그 상세한 설명을 생략한다.

[0019] 도 1은 본 발명의 일 실시예에 따른 레코드 복원방법의 흐름도이다.

[0020] 본 발명의 일 실시예에 따른 레코드 복원방법은 테이블 정보와 레코드 정보를 이용하여 삭제된 레코드를 추출하여 레코드를 복원한다. 특히, 삭제된 MySQL MyISAM 데이터베이스에서 삭제된 데이터를 복구하는데 이용될 수 있다. MySQL MyISAM 데이터베이스 5.5, 5.6 버전에서 적용이 가능하다.

[0021] MySQL MyISAM 데이터베이스는 테이블 구조 파일(FRM), 인덱스 파일(MYI), 데이터 파일(MYD) 3개의 파일을 통해 테이블을 관리한다. 테이블 구조 파일에는 테이블의 전반적인 스키마와 테이블의 타입이 저장된다. 테이블의 타입은 정적(Static) 테이블과 동적(Dynamic) 테이블이 존재한다. 정적 테이블의 경우 각 컬럼 데이터의 레코드 내 오프셋 또한 저장된다. 인덱스 파일에는 삭제 레코드의 개수와 마지막으로 삭제된 레코드의 위치가 저장된다. 데이터 파일은 실제 레코드가 저장되는 파일이다. MySQL MyISAM 데이터베이스 내에서 모든 삭제된 레코드는 Linked List 형식으로 서로 이어져있다. 이러한 특성과 테이블 구조 파일 및 인덱스 파일을 이용하여 데이터 파일에서 MySQL MyISAM 데이터베이스의 삭제된 레코드를 복원할 수 있다.

[0022] 삭제된 레코드를 복원하는 방법은 이하 자세히 설명하도록 한다.

[0023] 110 단계는 테이블 구조 파일(FRM)로부터 테이블 정보를 독출하고, 인덱스 파일(MYI)로부터 레코드 정보를 독출하는 단계이다.

[0024] 보다 구체적으로, 삭제된 레코드를 복원하기 위하여 우선 테이블 구조 파일로부터 테이블 정보를 독출하고, 인덱스 파일로부터 레코드 정보를 독출한다.

[0025] 테이블 구조 파일(FRM)의 구조는 도 6과 같다. 테이블 구조 파일은 Header, Key & Index Area, Column Metadata Area로 구성되어 있다. Header에서 테이블의 타입을 획득할 수 있다. Column Metadata Area에서는 테이블의 스키마를 확인할 수 있는 각 컬럼의 메타데이터를 획득할 수 있다. 표 1은 FRM 파일에서 획득할 수 있는 정보를 나타낸다.

표 1

[0026]

구성요소	설명
Storage Engine Type	테이블을 구성한 스토리지 엔진의 종류
Table Type	MyISAM 엔진의 경우 Static,Dynamic이 존재한다
Table Schema	컬럼의 개수, 각 컬럼의 타입, 컬럼 데이터의 오프셋 등

[0027] Table Type은 테이블 구조 파일의 오프셋 0x1E에 위치한 2바이트의 값이다. 이 값이 짝수일 경우 정적(Static) 테이블을 의미하며, 홀수일 경우 동적(Dynamic) 테이블을 의미한다. 도 7에서 위의 테이블의 경우 동적 테이블임을, 아래 테이블의 경우 정적 테이블임을 알 수 있다.

[0028] 도 8은 테이블의 스키마를 확인할 수 있는 컬럼 메타데이터의 구조를 나타낸 그림이다. 컬럼 메타데이터는 17바이트의 크기를 가지며, 컬럼의 개수만큼 테이블 구조 파일의 Column Metadata Area에 연속적으로 위치한다. 오프셋 0x05의 3바이트는 정적 테이블에서 필요한 해당 컬럼의 레코드 내 오프셋이 저장된다. 오프셋 0x03의 2바이트와 오프셋 0x09의 1바이트의 값으로 decimal 타입의 크기를 계산할 수 있다. 오프셋 0x0D의 1바이트와 0x0E의 1바이트로 해당 컬럼의 데이터 타입을 알 수 있다.

[0029] 인덱스 파일에서 획득할 수 있는 정보는 다음 표 2와 같다.

표 2

[0030]

구성요소	설명
Deleted Record Num	삭제된 레코드의 개수
Last Del Record Offset	마지막으로 삭제된 레코드의 오프셋

[0031] 120 단계는 상기 테이블 정보로부터 테이블 타입을 구분하는 단계이다.

[0032] 보다 구체적으로, 110 단계에서 독출한 테이블 정보로부터 테이블 타입이 정적(Static) 테이블인지 동적(Dynamic) 테이블인지를 구분한다. 앞에서 설명한 바와 같이, 테이블 구조 파일의 오프셋 0x1E에 위치한 2 바이트의 값이 짝수일 때 정적 테이블로 구분하고, 홀수일 경우 동적 테이블로 구분한다.

[0033] 130 단계는 상기 테이블 타입이 정적 테이블인 경우, 상기 테이블 정보로부터 각 컬럼의 레코드 내 오프셋을 획득하여 삭제된 레코드를 복원하는 단계이다.

[0034] 보다 구체적으로, 정적(Static) 테이블의 모든 레코드의 크기와 각 컬럼의 오프셋은 동일하다. 따라서, 테이블 정보로부터 각 컬럼의 레코드 내 오프셋을 획득하고, 이를 이용하여 삭제된 레코드를 복원할 수 있다.

[0035] 정적 테이블에서 레코드가 삭제될 시 레코드의 앞 7바이트가 덮어쓰인다. 덮어쓰이는 영역의 구조는 도 9와 같다. 삭제된 레코드의 Header는 1바이트 크기를 가지며, 항상 0의 값을 가진다. Header 다음 6바이트는 이전에 삭제된 레코드의 인덱스 번호가 저장되는데, 첫 번째로 삭제된 레코드의 경우에는 이전에 삭제된 레코드가 존재하지 않으므로 6바이트가 “0xFF” 값으로 채워진다. 인덱스 번호는 첫 번째 레코드를 뜻하는 “0x00”에서 시작하여 순차적으로 증가한다.

[0036] 140 단계는 상기 테이블 타입이 동적 테이블인 경우, 피봇을 설정하고, 컬럼 데이터 개수의 총합, 피봇, 및 레코드의 사이즈를 이용하여 각 컬럼의 레코드 내 오프셋을 도출하여 삭제된 레코드를 복원하는 단계이다.

[0037] 보다 구체적으로, 동적(Dynamic) 테이블의 모든 레코드의 크기와 각 컬럼의 오프셋은 다를 수 있다. 따라서, 각 컬럼의 오프셋을 찾아야만 레코드를 복원할 수 있다. 이를 위하여, 첫 번째 컬럼의 레코드 내 시작 오프셋으로 가정되는 오프셋인 피봇을 설정하고, 컬럼 데이터 개수의 총합, 피봇, 및 레코드의 사이즈를 이용하여 각 컬럼의 레코드 내 오프셋을 도출함으로써 삭제된 레코드를 복원한다.

[0038] 우선, 동적 테이블의 구조를 구체적으로 설명하면 다음과 같다.

[0039] 동적(Dynamic) 테이블의 모든 레코드의 크기와 각 컬럼의 오프셋은 다를 수 있고, 13종류의 레코드의 헤더가 존재하는데, 헤더의 종류에 따라 그 크기가 다르다. 헤더의 크기는 3바이트에서 20바이트 사이이다. 레코드가 삭제될 시 레코드의 앞 20바이트가 덮어쓰인다. 덮어쓰이는 영역의 구조는 도 10과 같다.

[0040] 삭제된 레코드의 Header는 1바이트 크기를 가지며, 항상 0의 값을 가진다. Block Length는 삭제된 레코드의 크기를 뜻한다. Prev_filepos는 이전에 삭제된 레코드의 시작 오프셋 정보이 저장되며, 이전에 삭제된 레코드가 존재하지 않으면 “0xFF” 값으로 채워진다. Next_filepos는 다음에 삭제된 레코드의 시작 오프셋 이 저장되며, 다음에 삭제된 레코드가 존재하지 않으면 “0xFF” 값으로 채워진다.

[0041] 동적 테이블의 복구는 데이터 타입에 따른 레코드 크기를 파악하는 것이 중요하다. 표 3은 데이터 타입 별 고유 값과 크기를 나타낸다.

표 3

[0042]

데이터 타입	값	데이터 크기	Collation
char	0xFE	가변길이	0x21
binary	0xFE	가변길이	0x3F
tinyint	0x01	1 byte	
smallint, bool	0x02	2 byte	
mediumint	0x09	3 byte	
int	0x03	4 byte	
bigint	0x08	8 byte	
float	0x04	4 byte	
double, real	0x05	8 byte	
decimal(a, b)	0xF6	decimal 크기 계산법 참조	

date	0x0E	3 byte	
datetime	0x12	5 byte	
time	0x13	3 byte	
timestamp	0x11	4 byte	
year	0x0D	1 byte	
set	0xF8	n-byte 8 멤버 이하 : 1 byte 16 멤버 이하 : 2 byte 24 멤버 이하 : 3 byte 32 멤버 이하 : 4 byte 32 멤버 초과 : 8 byte	
enum	0xF7	n-byte 255 멤버 이하 : 1byte 255 멤버 초과 : 2byte	
varchar	0x0F	가변길이	0x21
varbinary	0x0F	가변길이	0x3F
tinyblob	0xF9	가변길이	0x3F
tinytext	0xF9	가변길이	0x21
blob	0xFC	가변길이	0x3F
text	0xFC	가변길이	0x21
mediumblob	0xFA	가변길이	0x3F
mediumtext	0xFA	가변길이	0x21
longblob	0xFB	가변길이	0x3F
longtext	0xFB	가변길이	0x21

[0043] 동적 테이블에서 char와 binary 타입을 사용할 경우 자동으로 varchar와 varbinary 타입으로 변환된다.

[0044] Decimal 타입은 해당 컬럼의 메타데이터를 분석하여 데이터에 소수점 존재 여부를 확인한 뒤, 크기를 구할 수 있다. 컬럼 메타데이터의 오프셋 0x09의 1바이트 값이 0x80이거나 0x40이면 데이터에 소수점이 존재하지 않는 것을 의미하고, 그 외의 값은 소수점이 존재하는 것을 의미한다. 소수점이 존재할 경우 표 4와 같이 정수부와 소수부로 나누어 각각 크기를 계산하고, 소수점이 존재하지 않는 경우에는 정수부만 계산한다. 표 4에서 컬럼 메타데이터 내 오프셋 0x03의 2바이트(Number of byte in Column)의 값은 “N” 이라 하고, 오프셋 0x09의 값(Pack Flag[1])의 값은 “A”로, 정수부의 최대 자리수는 “N_{max}”, 소수부의 최대 자리수는 “D_{max}”라고 하였다.

표 4

[0045]

소수점 유무	옵션	정수부(I)	소수부(D)
무	최대 자리수	$N-1$	-
	데이터 크기	$*N_{max}/9+*4+((\log_2(N_{max}\%9)+1)/8)$	-
유	최대 자리수	$N-2-(A\ \&\ 0x3F)$	$A\ AND\ 0x3F$
	데이터 크기	$*N_{max}/9+*4+((\log_2(10^{N_{max}\bmod 9}-1)+1)/8)$	$*D_{max}/9+*4+((\log_2(10^{D_{max}\bmod 9}-1)+1)/8)$

[0046] 여기서, $*\ +$ 는 내림함수, $(\ )$ 는 올림함수, mod는 모듈러 함수(a mod n = r 즉, a를 n으로 나눈 나머지)를 뜻한다.

[0047] 가변길이의 데이터 타입은 레코드에 데이터가 저장될 때, 데이터의 크기를 나타내는 Length Byte가 데이터의 최상단에 위치한다. 표 5는 가변길이 데이터 타입 별 Length Byte의 크기를 나타낸다.

표 5

[0048]

데이터 타입	길이 (Byte)	해석 방법
varchar(n) varbinary(n)	컬럼에 할당 가능한 바이트 수(Number of bytes in Column) 255 이하 : 1 바이트 255 초과 : 2 바이트	빅 엔디언
tinytext tinyblob	1 바이트	리틀 엔디언
text blob	2 바이트	리틀 엔디언
mediumtext mediumblob	3 바이트	리틀 엔디언
longtext longblob	4 바이트	리틀 엔디언

[0049] 테이블 타입이 동적 테이블인 경우, 각 컬럼의 레코드 내 오프셋을 도출하여 레코드를 복원하는 단계를 도 2 내지 4를 참조하여 보다 구체적으로 설명하도록 한다.

[0050] 먼저, 210 단계에서 20번째 바이트를 피봇으로 설정한다.

[0051] 보다 구체적으로, 동적 테이블의 경우, 레코드 삭제 시 첫 20바이트가 삭제에 관련된 정보로 덮어 쓰인다. 동적 테이블은 첫 번째 컬럼의 데이터 오프셋을 알 수 없으므로 20번째 바이트가 첫 번째 컬럼의 데이터 시작 오프셋이라 가정하고 이를 피봇이라 정의한다. 또한, 모든 컬럼의 데이터가 20바이트에 덮어쓰이지 않았다고 가정한다.

[0052] 210 단계에서 피봇을 설정하고, 220 단계에서 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은지 판단한다.

[0053] 보다 구체적으로, 각 컬럼 데이터 개수의 총합에 피봇을 더한 값이 삭제된 레코드의 크기와 같은지를 비교한다. 초기 피봇은 20번째 바이트가 설정되어 있는바, 초기 피봇은 20이다.

[0054] 220 단계의 판단 결과, 상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은 경우, 현재 피봇을 첫 번째 컬럼의 레코드 내 시작 오프셋으로 도출하고 삭제된 레코드를 복원한다.

[0055] 보다 구체적으로, 컬럼 데이터 개수의 총합에 피봇을 더한 값이 삭제된 레코드의 크기와 같다면, 현재 설정한 컬럼 데이터의 오프셋이 맞다고 판단하고 각 컬럼의 레코드 내 오프셋을 이용하여 레코드를 복구한다.

[0056] 220 단계의 판단 결과, 상기 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 다른 경우, 240 단계와 같이, 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같아질 때까지 피봇을 1씩 증가시키면서 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복한다.

[0057] 보다 구체적으로, 컬럼 데이터 개수의 총합에 피봇을 더한 값이 삭제된 레코드의 크기와 같지 않다면, 피봇으로 설정된 바이트는 첫 번째 컬럼의 레코드 내 오프셋이 아니고, 피봇 이후에 위치하는 오프셋이 첫 번째 컬럼의 레코드 내 오프셋일 수 있는바, 피봇을 1 증가시킨 후, 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복한다. 즉, 피봇을 20번째 바이트에서 21 번째 바이트로 이동시키고, 220 단계에서 컬럼 데이터 개수의 총합과 피봇의 합이 삭제된 레코드의 사이즈와 같은지 판단한다.

[0058] 다만, 현재 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같은 경우, 더이상 피봇을 증가시킬 수 없게되고, 현재 컬럼의 수만큼 복원할 수 있는 레코드의 수가 부족하다는 것을 의미하는바, 310 단계에서 상기 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같은지 판단한다.

[0059] 310 단계의 판단 결과, 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같지 않은 경우, 아직 피봇을 증가시키면서 판단할 경우가 남아있는바, 320 단계에서 피봇을 1 증가시키고, 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복한다.

[0060] 이와 다르게 310 단계의 판단 결과, 피봇에 1을 더한 값이 삭제된 레코드의 사이즈와 같은 경우, 복원할 수 있는 컬럼의 수가 부족한바, 330 단계에서 복원할 수 있는 컬럼의 수가 0이 될 때까지 복원할 수 있는 컬럼의 수를 한 개씩 줄이면서 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복한다. 컬럼의 수를 한 개씩 줄이는 경우, 피봇은 다시 20번째 바이트를 피봇으로 설정하고, 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복한다.

[0061] 330 단계에서 복원할 수 있는 컬럼의 수를 줄여나가다 상기 복원할 수 있는 컬럼의 수가 0인 경우, 더 이상 복원할 수 있는 컬럼의 수가 없는 결과가 도출될 수 있다. 하지만, 동적 테이블은 동적 테이블에서 컬럼의 데이터가 데이터 파일에 저장되지 않는 스킵(Skip)이 가능한 데이터 타입이 있다. 컬럼의 데이터가 레코드에 저장되지 않고 스킵 될 때, 레코드의 헤더 뒤에 해당 컬럼의 데이터가 스킵 되었다고 1비트 크기로 Bitmap을 표시해둔다. 이를 Bitmap Flag라고 한다. 동적 테이블의 레코드가 삭제되면 20바이트가 덮어쓰이므로 Flag Byte의 위치와 값을 알 수 없다. 그러므로 테이블 내 스킵이 가능한 컬럼 타입을 파악한 후 모든 가능성에 대하여 각 컬럼의 크기의 합을 구한 뒤, 삭제된 레코드 크기와 비교하여 각 컬럼의 오프셋을 찾아야 한다. 즉, 테이블 정보로부터 스킵이 가능한 컬럼이 있는지 확인하고, 스킵이 가능한 컬럼이 있는 경우, 스킵이 가능한 컬럼을 고려하여 상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복한다.

[0062] 표 6은 Bitmap Flag에 반영되지 않는 예외 타입을 나타낸다. 다음 데이터 타입의 컬럼 데이터는 데이터 파일에 항상 저장되어야 하며, 스킵이 될 수 없다. 이외의 데이터 타입의 경우, 스킵이 가능한바, 이를 고려하여 상기 각 컬럼의 레코드 내 오프셋을 도출하여야 한다.

표 6

[0063]

데이터 타입
varchar
varbinary
decimal
datetime
time
timestamp

[0064] 상기 레코드 정보에 포함된 삭제된 레코드의 수 및 마지막으로 삭제된 레코드의 오프셋 정보를 이용하여 삭제된 모든 레코드에 대해 복원을 수행한다.

[0065] 보다 구체적으로, 인덱스 파일로부터 독출된 레코드 정보에 포함된 마지막으로 삭제된 레코드의 오프셋 정보를 이용하여 해당 레코드에 대한 복원이 완료된 경우, 삭제된 레코드에서 도출된 이전에 삭제된 레코드의 인덱스 정보가 포함되어 있는바, 이를 이용하여 이전에 삭제된 레코드에 대한 복원을 반복 수행하여, 삭제된 모든 레코드에 대한 복원을 수행한다. 레코드 정보에 포함된 삭제된 레코드의 수만큼 레코드 복원을 반복 수행한다.

[0066] 상기 과정들을 하나의 흐름도로 나타내면 도 5와 같이 나타낼 수 있다.

[0067] 먼저, 510 단계에서 데이터 구조 파일(FRM)로부터 테이블 정보를 독출하고, 인덱스 파일(MYI)로부터 레코드 정보를 독출한다. 삭제된 레코드가 있는지 확인하고, 삭제된 레코드가 있는 경우, 520 단계에서 테이블 타입을 구분한다. 정적(Static) 테이블인 경우(530), 각 컬럼의 레코드 내 오프셋이 정해져 있는바, 이 정보를 이용하여 각 컬럼 데이터를 복원함으로써 삭제된 레코드를 복원(532)한다. 하나의 레코드에 대한 복원이 완료되면 532 단계에서 이전에 삭제된 레코드가 있는지 확인하여 삭제된 모든 레코드에 대해 복원을 수행한다.

[0068] 테이블 타입이 동적(Dynamic) 테이블인 경우(540), 먼저, 541 단계에서 20번째 바이트를 피봇(Pivot)으로 설정하고, 542 단계에서 컬럼 데이터 개수의 총합과 피봇의 합이 레코드의 사이즈와 같은지 판단한다. 각 컬럼 길이의 총합과 피봇의 합이 레코드의 사이즈와 같은 경우, 현재, 피봇을 첫 번째 컬럼의 시작 오프셋으로 도출하고, 이에 따라 각 컬럼 데이터를 복원함으로써 삭제된 레코드를 복원(543)한다. 하나의 레코드에 대한 복원이 완료되면 544 단계에서 이전에 삭제된 레코드가 있는지 확인하여 삭제된 모든 레코드에 대해 복원을 수행한다.

[0069] 542 단계의 판단 결과, 각 컬럼 길이의 총합과 피봇의 합이 레코드의 사이즈와 같지 않은 경우, 551 단계에서 피봇을 증가시킬 필요가 있는지를 확인하기 위하여, 피봇값에 1을 더한 값이 레코드 사이즈와 같은지 판단한다. 피봇값에 1을 더한 값이 레코드 사이즈와 같지 않은 경우, 523 단계에서 피봇을 1 증가시킨다. 즉, 피봇값을 1 증가시킨다. 피봇값을 1 증가한 후, 새로운 피봇값을 이용하여 542 단계를 다시 수행한다.

[0070] 551 단계의 판단결과, 피봇값에 1을 더한 값이 레코드 사이즈와 같은 경우, 더이상 피봇을 증가시킬 필요가 없는바, 553 단계에서 복원할 수 있는 컬럼의 수를 하나 줄이고(Col=Col-1), 피봇값을 다시 20으로 설정한다. 하나 줄인 컬럼의 수가 0인지 554 단계에서 판단하여, 컬럼의 수가 0이 아닌 경우, 542 단계를 다시 수행한다. 하나 줄인 컬럼의 수가 0인 경우, 더이상 복원할 수 있는 컬럼의 수가 없는 것을 의미한다. 이때, 바로 복원을 종료하지 않고, 스킵이 가능한 컬럼이 있는지를 확인(555)하여, 컬럼의 스킵에 따라 컬럼의 수가 적었는지를 확인하고, 스킵이 가능한 컬럼이 있는 경우, 이를 고려하여, 스킵이 된 경우들에 대해 542 단계 다시 수행한다.

[0071] 최종 모든 삭제된 레코드에 대한 복원이 완료되거나(532, 544) 더이상 복원이 가능한 레코드가 없는 경우(555), 복원 과정을 종료한다.

[0072] 본 발명의 일 실시예에 따른 레코드 복원 장치는 하나 이상의 저장부와 하나 이상의 처리부로 구성될 수 있고, 외부와 데이터를 송수신하는 통신부를 더 포함할 수 있다. 저장부는 테이블 구조 파일, 인덱스 파일, 데이터 파일을 저장될 수 있고, 처리부에서의 처리 결과를 저장할 수 있다. 처리부는 테이블 구조 파일로부터 테이블 정보를 독출하고, 인덱스 파일로부터 레코드 정보를 독출하고, 상기 테이블 정보로부터 테이블 타입을 구분하며, 상기 테이블 타입이 정적 테이블인 경우, 상기 테이블 정보로부터 각 컬럼의 레코드 내 오프셋을 획득하여 삭제된 레코드를 복원하고, 상기 테이블 타입이 동적 테이블인 경우, 피봇을 설정하고, 컬럼 데이터 개수의 총합, 피봇, 및 레코드의 사이즈를 이용하여 각 컬럼의 레코드 내 오프셋을 도출하여 삭제된 레코드를 복원한다. 처리부에 수행되는 복원 과정에 대한 상세한 설명은 도 1 내지 도 10에 대한 레코드 복원 방법에 대한 상세한 설명에 대응한다.

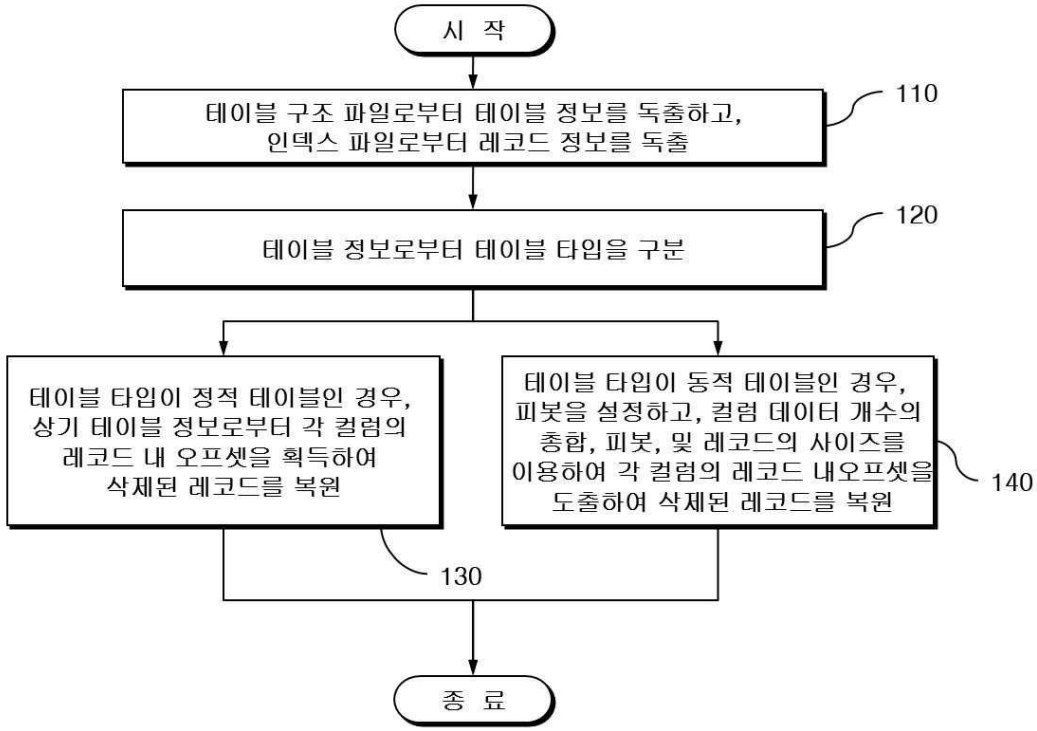
[0073] 본 발명의 실시예들은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 본 발명을 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체 (magnetic media), CD-ROM, DVD와 같은 광기록 매체 (optical media), 플롭티컬 디스크 (floptical disk)와 같은 자기-광 매체 (magneto-optical media), 및 롬 (ROM), 램 (RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 본 발명의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0074] 이상과 같이 본 발명에서는 구체적인 구성 요소 등과 같은 특정 사항들과 한정된 실시예 및 도면에 의해 설명되었으나 이는 본 발명의 보다 전반적인 이해를 돕기 위해서 제공된 것일 뿐, 본 발명은 상기의 실시예에 한정되는 것은 아니며, 본 발명이 속하는 분야에서 통상적인 지식을 가진 자라면 이러한 기재로부터 다양한 수정 및 변형이 가능하다.

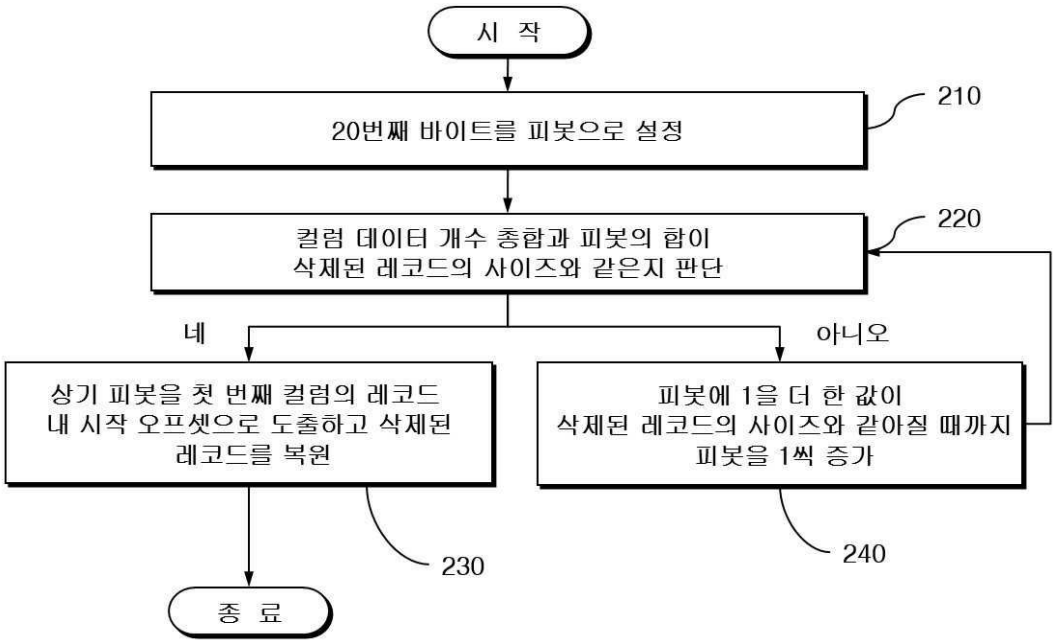
[0075] 따라서, 본 발명의 사상은 설명된 실시예에 국한되어 정해져서는 아니되며, 후술하는 특허청구범위뿐 아니라 이 특허청구범위와 균등하거나 등가적 변형이 있는 모든 것들은 본 발명 사상의 범주에 속한다고 할 것이다.

도면

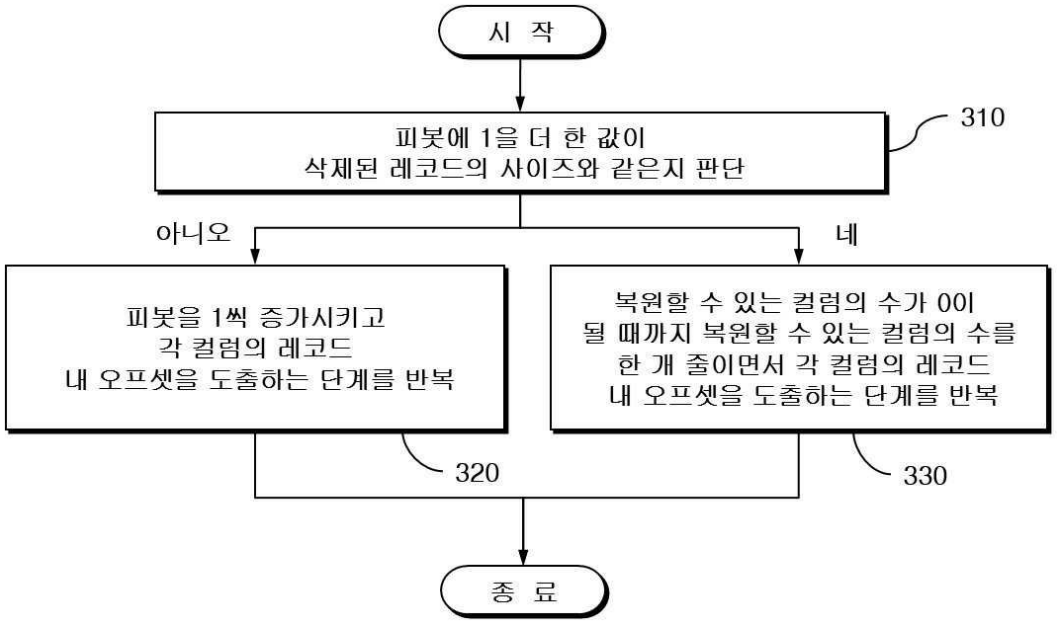
도면1


시 작
테이블 구조 파일로부터 테이블 정보를 독출하고,
인덱스 파일로부터 레코드 정보를 독출
110
테이블 정보로부터 테이블 타입을 구분
120
테이블 타입이 정적 테이블인 경우,
상기 테이블 정보로부터 각 컬럼의
레코드 내 오프셋을 획득하여
삭제된 레코드를 복원
테이블 타입이 동적 테이블인 경우,
피봇을 설정하고, 컬럼 데이터 개수의
총합, 피봇, 및 레코드의 사이즈를
이용하여 각 컬럼의 레코드 내오프셋을
도출하여 삭제된 레코드를 복원
140
130
종 료

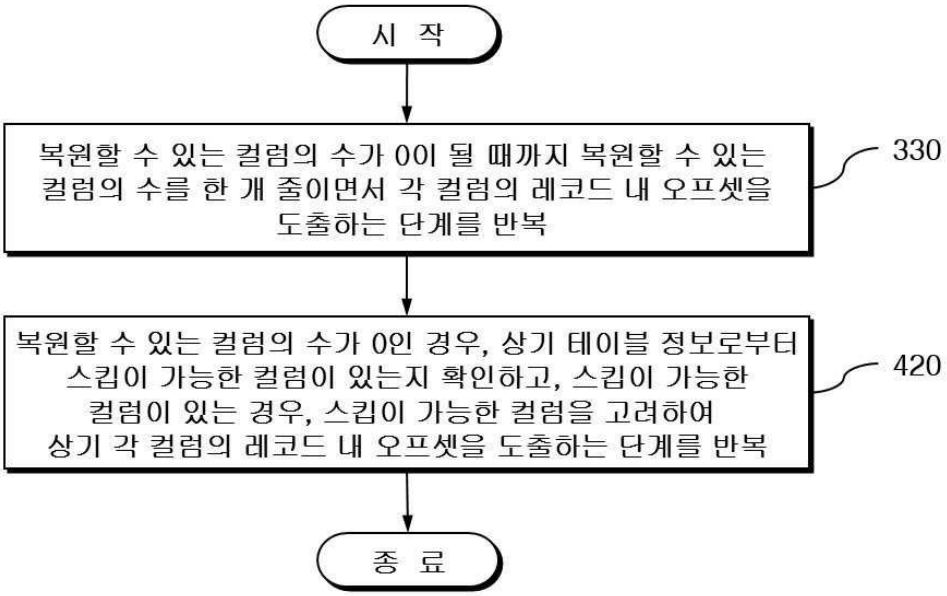
도면2


시 작
20번째 바이트를 피봇으로 설정
210
컬럼 데이터 개수 총합과 피봇의 합이
삭제된 레코드의 사이즈와 같은지 판단
220
네
아니오
상기 피봇을 첫 번째 컬럼의 레코드
내 시작 오프셋으로 도출하고 삭제된
레코드를 복원
피봇에 1을 더 한 값이
삭제된 레코드의 사이즈와 같아질 때까지
피봇을 1씩 증가
230
240
종 료

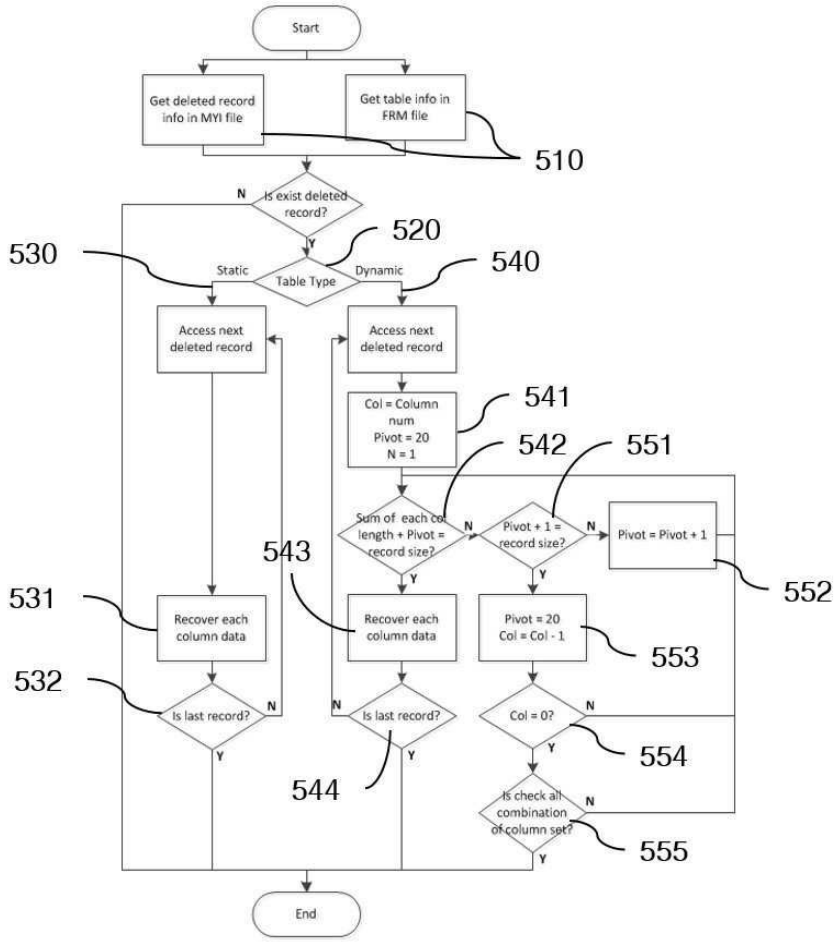
도면3


시 작
피봇에 1을 더 한 값이
삭제된 레코드의 사이즈와 같은지 판단
310
아니오
네
피봇을 1씩 증가시키고
각 컬럼의 레코드
내 오프셋을 도출하는 단계를 반복
320
복원할 수 있는 컬럼의 수가 0이
될 때까지 복원할 수 있는 컬럼의 수를
한 개 줄이면서 각 컬럼의 레코드
내 오프셋을 도출하는 단계를 반복
330
종 료

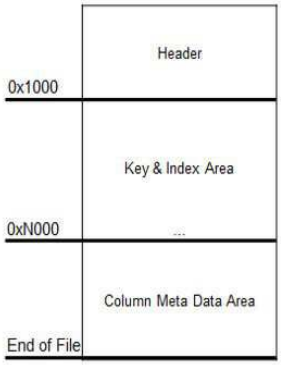
도면4


시 작
복원할 수 있는 컬럼의 수가 0이 될 때까지 복원할 수 있는
컬럼의 수를 한 개 줄이면서 각 컬럼의 레코드 내 오프셋을
도출하는 단계를 반복
330
복원할 수 있는 컬럼의 수가 0인 경우, 상기 테이블 정보로부터
스킵이 가능한 컬럼이 있는지 확인하고, 스킵이 가능한
컬럼이 있는 경우, 스킵이 가능한 컬럼을 고려하여
상기 각 컬럼의 레코드 내 오프셋을 도출하는 단계를 반복
420
종 료

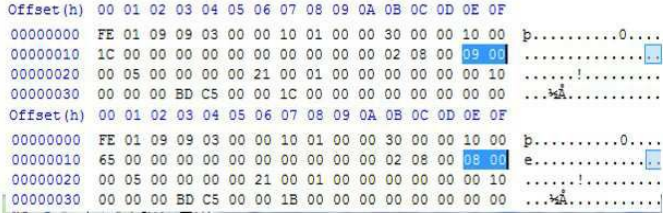
도면5


Start
Get deleted record info in MYI file
Get table info in FRM file
510
Is exist deleted record?
N
Y
520
530
Static
Table Type
Dynamic
540
Access next deleted record
Access next deleted record
Col = Column num
Pivot = 20
N = 1
541
542
551
Sum of each col length + Pivot = record size?
Pivot + 1 = record size?
Pivot = Pivot + 1
552
543
531
Recover each column data
Recover each column data
Pivot = 20
Col = Col - 1
553
532
Is last record?
Is last record?
Col = 0?
554
544
Is check all combination of column set?
555
End

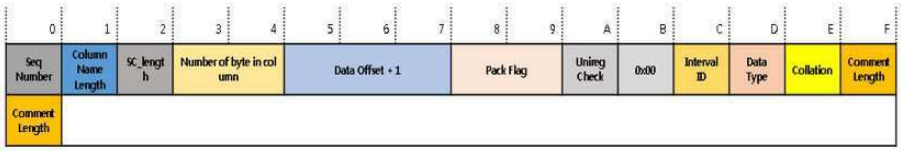
도면6


Header
0x1000
Key & Index Area
0xN000
...
Column Meta Data Area
End of File

도면7


Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 FE 01 09 09 03 00 00 10 01 00 00 30 00 00 10 00
00000010 1C 00 00 00 00 00 00 00 00 00 00 02 08 00 09 00
00000020 00 05 00 00 00 00 21 00 01 00 00 00 00 00 00 10
00000030 00 00 00 BD C5 00 00 1C 00 00 00 00 00 00 00 00
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 FE 01 09 09 03 00 00 10 01 00 00 30 00 00 10 00
00000010 65 00 00 00 00 00 00 00 00 00 00 02 08 00 08 00
00000020 00 05 00 00 00 00 21 00 01 00 00 00 00 00 00 10
00000030 00 00 00 BD C5 00 00 1B 00 00 00 00 00 00 00 00

도면8


0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
Seq Number
Column Name Length
SC_length
Number of byte in column
Data Offset + 1
Pack Flag
Unireg Check
0x00
Interval ID
Data Type
Collation
Comment Length
Comment Length

도면9


Header
(1 Byte)
Previous Deleted Record's Index
(6 Byte)

도면10


Header Signature
(1 Byte)
Block Length
(3 Byte)
Prev_filepos
(8 Byte)
Next_filepos
(8 Byte)